TUGAS AKHIR – KI141502

IMPLEMENTASI MIDI FILE HANDLER DALAM *RHYTHM GAME* YANG BERSIFAT OTOMATIS

ANTONIUS STANLEY LIMANTO
NRP 5113100088

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Anny Yuniarti, S.Kom, M.Comp.Sc.

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2018

TUGAS AKHIR – KI141502

IMPLEMENTASI MIDI FILE HANDLER DALAM *RHYTHM GAME* YANG BERSIFAT OTOMATIS

ANTONIUS STANLEY LIMANTO
NRP 5113100088

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Anny Yuniarti, S.Kom, M.Comp.Sc.

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

(Halaman Ini Sengaja dikosongkan)

FINAL PROJECT– KI141502

MIDI FILE HANDLER IMPLEMENTATION IN AN AUTOMATIC RHYTHM GAME

ANTONIUS STANLEY LIMANTO
NRP 5113100088

Advisor
Imam Kuswardayan, S.Kom., M.T.
Anny Yuniarti, S.Kom, M.Comp.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
TENTH NOVEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2018

(Halaman Ini Sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI METODE MIDI FILE HANDLER DALAM *RHYTHM GAME* YANG BERSIFAT OTOMATIS

Tugas Akhir

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Interaksi, Grafika, dan Seni
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi Dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
ANTONIUS STANLEY LIMANTO
NRP. 5113100088

Disetujui oleh Dosen Pembimbing Tugas Akhir:
Imam Kuswardayan, S.Kom., M.T. .....
NIP: 19761215 200312 1 001 (pembimbing 1)

Anny Yuniarti, S.Kom, M.Comp.Sc. .....
NIP: 198106222005012002 (pembimbing 2)

SURABAYA
JANUARI, 2018

(Halaman Ini Sengaja dikosongkan)

IMPLEMENTASI METODE MIDI FILE HANDLER DALAM *RHYTHM GAME* YANG BERSIFAT OTOMATIS

Nama Mahasiswa : Antonius Stanley Limanto
NRP : 5113100088
Jurusan : Departemen Informatika FTIK-ITS
Dosen Pembimbing I : Imam Kuswardayan, S.Kom., M.T.
Dosen Pembimbing II : Anny Yuniarti, S.Kom, M.Comp.Sc.

ABSTRAK

Rhythm Game sedang *booming* saat ini. Tidak hanya pada media *arcade* saja, namun sekarang sudah dapat diimplementasikan untuk versi dirumah dan media portabel. Tiap game tersebut memiliki koleksi lagu yang berbeda-beda, bisa dari game tersebut yang menyediakan lagu dari game itu sendiri maupun melisensi lagu lain. Game ini memiliki unsur yang menarik banyak orang dikarenakan oleh lagu yang dikoleksinya. Saat ini *Rhythm Game* selalu membutuhkan seorang *note designer* untuk membuat *note mapping* tersebut, lagu yang sudah dibuat tidak akan membuat *note* dengan sendirinya.

Dalam Tugas Akhir ini dibangun suatu permainan yang bergenre *Rhythm Game*. *Rhythm Game* yang akan dibuat dalam Tugas Akhir ini akan menggunakan lagu midi lalu mengimplementasikan Midi File Handler di dalam sistem permainan. Midi File Handler ini digunakan untuk membaca file midi, dalam pembacaan file tersebut Midi File Handler melakukan *sequencing* sehingga menghasilkan suatu informasi pada *midi event*. *Midi event* ini akan berisi suatu informasi singkat berupa, *channel*, *note (*nada keberapa yang akan dibunyikan) serta *velocity* atau amplitudo suara. Ketiga informasi inilah yang akan menentukan kapan atau yang mana suatu *note* akan dibunyikan saat lagu dimainkan. Midi File Handler ini telah digunakan pada beberapa lagu midi (atau file .mid) dan terbukti dapat mendeteksi *midi event* dengan lancer dan pada *timing* yang pas.

Dalam Tugas Akhir ini akan menggunakan beberapa lagu midi yang bersifat *ideal* yang kebanyakan adalah lagu Indonesia, dalam bentuk konversi .mp3/.wav yang digunakan untuk media suara pada permainan nanti serta dalam ekstensi .bytes yang akan digunakan untuk membaca *midi event* melalui Midi File Handler, berkat file .bytes inilah yang akan menghasilkan suatu *note*. Diantara file .bytes serta file .mp3/.wav akan ada waktu tunda agar *note* yang dibuat dapat pas sesuai file .mp3/.wav. Hasil uji coba pada Tugas Akhir ini akan berupa persentase keberhasilan meliputi aspek konsistensi, kedinamisan dan ketepatan *timing* pada Midi File Handler.

Kata kunci: ***channel*, midi, Midi File Handler, *note*, otomatis, *Rhythm Game*, *velocity*.**

MIDI FILE HANDLER IMPLEMENTATION IN AN AUTOMATIC RHYTHM GAME

Student Name	: Antonius Stanley Limanto
NRP	: 5113100088
Major	: Informatics Department FTIK-ITS
Advisor I	: Imam Kuswardayan, S.Kom., M.T.
Advisor II	: Anny Yuniarti, S.Kom, M.Comp.Sc.

ABSTRACT

These days, Rhythm Game is really booming. Not only for arcades but now they have been implemented the home version and portables. Every each of that games have different collections of songs, they can be from their own original compotions or licensing other songs. This type of game have been attracting many people because of the songs the games collect. These days Rhythm Games always hire note designers to make a note mapping in songs, it's because the song can't produce the notes itself.

So in this Final Project, will be constructed a Rhythm Game application, that will be applying a Midi File Handler in the game. This will be used as a midi file reader that will produce a set of information called midi event. Midi event will contain data such as, channel, note, and velocity, these three information will decide when and which note will be produced. This Midi File Handler has been tested with some .mid files and it's proven its ability to detect midi event fluently with a perfect timing.

In this Final Project will be used some ideal midi files many of them are such as Indonesian songs, they will be converted in .mp3/.wav file that will be used for audio media and will be extended as .bytes file for midi event reading by Midi File Handler. Between .mp3/.wav files and .bytes files there will be a delaying time configured. So that the .mp3/.wav file will play in a perfect timing with the notes produced by the .bytes file. The result of this experiment will be percentages of Midi File

Handler's successfulness such as consistency, dynamicity dan timing precision.
Keywords:** : **automatic, channel, midi, Midi File Handler, note, Rhythm Game, velocity.

KATA PENGANTAR

Segala puji dan syukur kepada Tuhan Yang Maha Esa yang telah memberikan berkat, karunia dan bimbingannya sehingga penulis dapat menyelesaikan tugas akhir dengan judul "IMPLEMENTASI METODE MIDI FILE HANDLER DALAM *RHYTHM GAME* YANG BERSIFAT OTOMATIS".

Pengerjaan tugas akhir ini adalah kesempatan bagi penulis untuk merepresentasikan kemampuan, keinginan serta tujuan pribadi terhadap pembelajaran selama selama 8-9 semester di kampus Departemen Informatika Institut Teknologi Sepuluh Nopember Surabaya.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan-bantuan yang penulis terima dari berbagai pihak. Melalui lembar ini, penulis ingin secara khusus menyampaikan ucapan terima kasih kepada:

1. Tuhan yang Maha Esa atas bimbingan, berkat secara fisik, mental dan emosional serta utusannya kepada roh kudus dalam menuntun penulis mengerjakan dalam Tugas Akhir.
2. Keluarga yang sudah 22 tahun menemani penulis, memelihara penulis serta mendoakan penulis dengan penuh kasih sayang, khususnya Ayah Penulis atas nama Andreas Rudjujanto serta Ibu Penulis atas nama Lidyawati Ham.
3. Bapak Imam Kuswardayan selaku dosen pembimbing Tugas Akhir pertama yang telah memberikan arahan dalam pengerjaan Tugas Akhir ini.
4. Ibu Anny Yuniarti selaku dosen pembimbing Tugas Akhir kedua yang dengan sabar membimbing penulis dalam pengerjaan Tugas Akhir ini.
5. Ibu Bilqis Amaliah selaku dosen wali tahun awal yaitu semester satu hingga tujuh yang telah memberikan arahan dan menjadi sosok yang memberikan solusi dengan baik terhadap penulis.
6. Bapak Imam Kuswardayan selaku dosen wali semester akhir yang berkenan membantu dalam memecahkan permasalahan

terhadap semester akhir dimana penulis sedang dalam masalah.

7. Bapak Radityo Anggoro selaku dosen koordinator Tugas Akhir yang telah membantu penulis atas segala sesuatu mengenai syarat-syarat dan terlaksananya sidang Tugas Akhir.
8. Bapak Darlis Herumurti selaku Ketua Departemen Informatika ITS yang selama ini memberikan bantuan kepada penulis.
9. Dosen-dosen Departemen Informatika yang dengan sabar mendidik dan memberikan ilmu keahlian baik secara ilmu akademik maupun afektif kepada penulis selama di Departemen Informatika.
10. Staf TU Departemen Informatika ITS yang senantiasa sedia untuk membantu segala urusan penulis di jurusan.
11. Rekan-rekan dan pengelola Laboratorium Interaksi, Grafik, dan Seni yang telah memberikan fasilitas dan kesempatan melakukan riset atas Tugas Akhir yang dikerjakan penulis.
12. Rekan-rekan dan sahabat-sahabat penulis angkatan 2013 yang sama-sama memberikan dukungan baik secara langsung maupun tidak langsung (motivasi) kepada penulis.
13. Pihak-pihak lain yang tidak sengaja terlewat dan tidak dapat penulis sebutkan satu per satu.

Penulis telah berusaha sebaik mungkin dalam menyusun Tugas Akhir ini, namun mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Januari 2018

Antonius Stanley Limanto

DAFTAR ISI

(Halaman Ini Sengaja Dikosongkan)

DAFTAR GAMBAR

(Halaman Ini Sengaja Dikosongkan)

DAFTAR TABEL

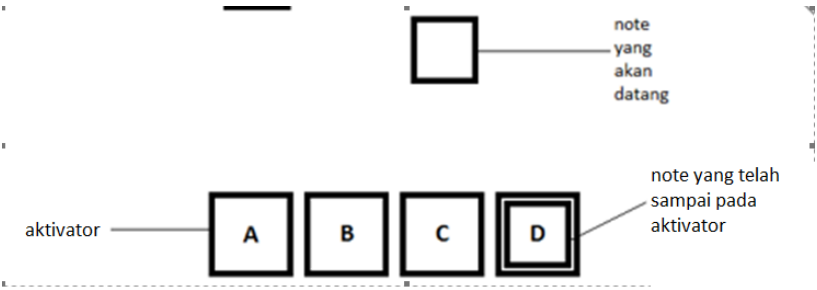
BAB I
PENDAHULUAN

Bab ini akan berisi pendahuluan yang akan menjelaskan garis besar dan maksud dari pembuatan tugas akhir ini, akan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan metodologi.

1.1 Latar Belakang

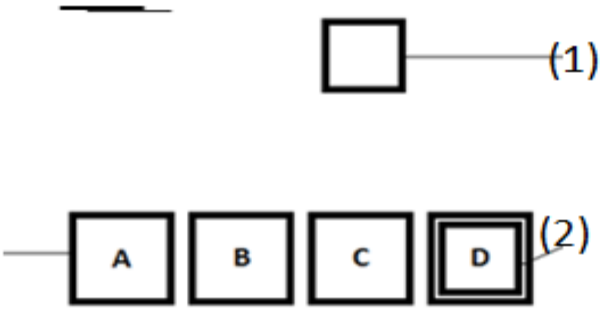
Rhythm Game atau disebut juga dengan permainan musik adalah aliran dari permainan yang mengasah kemampuan pemain untuk menganalisa beat dari suatu lagu. *Rhythm Game* adalah permainan yang sangat mudah untuk dipahami cara bermainnya dan digemari oleh banyak orang karena pada dasarnya semua orang menyukai musik. Di negara lain pun, sudah banyak industri games yang sudah memproduksi *Rhythm Game* dengan bentuk yang berbeda-beda, bentuk distribusinya pun bermacam-macam, dari home console, PC (Windows, Linux), mobile (iOS, Android) maupun arcade (Timezone, Amazone).

Rhythm Game jenisnya bermacam-macam inputnya ada yang melalui tombol, touch screen dan sebagainya . Namun struktur dasar dari *Rhythm Game* tetap sama yaitu terdiri dari *note* dengan aktivator seperti yang diilustrasikan pada **Gambar 1.1**.



Gambar 1.1 Ilustrasi Struktur Dasar pada *Rhythm Game*

dan cara bermainnya pun cukup sederhana yaitu pemain memberikan input disaat *note* sudah mencapai aktivator. Seperti yang terlihat pada **Gambar 1.2**.



Gambar 1.2 Cara memainkan *Rhythm Game*

Pada gambar di atas, dasarnya cara memainkan *Rhythm Game* ialah:
1. Menunggu hingga *note* sampai menuju aktivator.
2. Melakukan input yang sesuai apabila *note* telah menuju aktivator.

Memang sudah banyak industri game atau perorangan yang telah membuat *Rhythm Game*. Tetapi *Rhythm Game* yang telah dibuat bersifat tidak otomatis, sehingga untuk proses penyusunan *note*nya, diperlukan seorang *note designer* yang diluar pihak programmer untuk menyusun/melakukan *mapping note* tiap lagu. Oleh karena itu, dengan adanya *library* pemrograman untuk *midi event* pada suatu file midi, munculah ide untuk membuat *Rhythm Game* yang otomatis, dimana *note* akan disusun sendiri dengan algoritma tertentu. Dengan ini tujuan tugas akhir ini adalah untuk membuat suatu *Rhythm Game* yang sudah tidak memerlukan *note designer* untuk mengisi *note* pada suatu lagu.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana aturan main, skenario dan tingkat kesulitan dalam game?
2. Bagaimana sistem membaca file midi sehingga mendapatkan data *midi event*, sehingga tahu kapan suatu *note* akan dibunyikan?
3. Bagaimana integrasi ketukan dalam permainan?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. File lagu yang akan digunakan hanyalah file standard midi atau SMF yang menggunakan *general midi event*, salah satunya adalah lagu Indonesia khususnya lagu daerah, dalam bentuk file .mp3/.wav untuk media suara dan file .bytes sebagai file midi yang akan diproses.
2. Input pada game hanyalah berupa mouse serta keyboard A, S, D, J , K, L.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah untuk membangun suatu sistem *Rhythm Game* yang mengimplementasikan Midi File Handler untuk mendapatkan data *midi event* yang digunakan untuk membuat *note* secara otomatis sehingga *Rhythm Game* tersebut bersifat otomatis.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini antara lain:

1. Memberikan media hiburan bagi para pengguna.
2. Untuk mengetahui penerapan Midi File Handler dalam suatu *Rhythm Game* dimana Midi File Handler tersebut akan mengekstraksi *midi event* dari file midi akan berisi informasi

yang dapat dimanfaatkan dalam *Rhythm Game* untuk menghasilkan suatu *note*.

1.6 Metodologi

Pembuatan Tugas Akhir dilakukan dengan menggunakan metodologi sebagai berikut:

A. Studi literatur
   Tahap Studi Literatur merupakan tahap pembelajaran dan pengumpulan informasi yang digunakan untuk mengimplementasikan Tugas Akhir. Tahap ini diawali dengan pengumpulan literatur, eksplorasi teknologi dan pustaka, serta pemahaman dasar teori yang digunakan pada topik Tugas Akhir. Literatur-literatur yang dimaksud disebutkan sebagai berikut:
   1. *Rhythm Game*
   2. Midi
   3. *Beats per Minute*
   4. Midi File Handler
   5. SMFLite (*Midi toolkit* yang akan menjadi Midi File Handler)
   6. Unity

B. Perancangan perangkat lunak
   Pada tahap ini dilakukan analisa awal dan pendefinisian kebutuhan sistem untuk mengetahui permasalahan yang sedang dihadapi. Selanjutnya, dirumuskan rancangan sistem yang dapat memberi solusi terhadap permasalahan tersebut. Langkah yang akan digunakan pada tahap ini adalah sebagai berikut:
   1. Pencarian file midi yang sesuai untuk sistem (yang bersifat ideal).
   2. Perancangan sistem dan mekanisme game.
   3. Perancangan relasi antara Midi File Handler dan file midi yang nantinya akan digunakan sebagai pembuatan *note* dalam permainan.

C. Implementasi dan pembuatan sistem
   Tahap implementasi merupakan tahap untuk membangun aplikasi permainan beserta sistem yang terkait. Aplikasi ini akan dibangun dengan bahasa pemrograman C# untuk Unity dengan IDE Unity Editor 2017.1.2f1. Aplikasi yang akan dibangun berbasis perangkat Windows PC.

D. Uji coba dan evaluasi
   Pada tahap ini akan dilakukan pengujian terhadap perangkat lunak menggunakan data atau skenario yang telah dipersiapkan sebelumnya. Evaluasi dapat berupa uji pada Midi File Handler yang telah diterapkan, apakah dalam *runtime*, Midi File Handler tersebut terdapat suatu kendala secara fungsional meliputi *note loss* atau sebagainya, atau secara performa meliputi kelancaran pada saat *runtime* apabila Midi File Handler tersebut menyebabkan berat pada komputer, *frame skipping*, atau ketidaktepatan *note* pada saat *runtime.*

E. Penyusunan laporan tugas akhir
   Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut.

- BAB I PENDAHULUAN

  Bab ini berisi latar belakang masalah, rumusan dan batasan permasalahan, tujuan dan manfaat pembuatan tugas akhir, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

- BAB II TINJAUAN PUSTAKA

Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir ini, tinjauan pustaka ini akan lebih menitik beratkan pada literatur yang berhubungan dengan *Rhythm Game* serta midi.

- BAB III ANALISIS DAN PERANCANGAN

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

- BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi *Note Trigger* yang memanfaatkan Midi File Handler, dan antarmuka permainan. Implementasi ini akan menggunakan Unity sebagai Editor UI serta Bahasa C# khusus Unity.

- BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat keluaran yang dihasilkan oleh aplikasi dan evaluasi untuk mengetahui kemampuan aplikasi.

- BAB VI PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir. Teori-teori tersebut meliputi *Rhythm Game*, midi, BPM (*Beats per Minute)*,Midi File Handler, SMFLite(C# Midi ToolKit), Unity.

2.1 *Rhythm Game*

Rhythm Game adalah permainan bergenre musik yang memberikan tantangan bagi pemainya untuk menganilisis, ketukan pada lagu. *Rhythm Game* pada umumnya memiliki struktur dasar yaitu musik, aktivator dan *note* lalu biasanya ditambah dengan sistem *scoring* yang variatif, aturan menang kalah dan sebagainya. Salah satu karakteristik yang membuat *Rhythm Game* menjadi menarik dan pemain betah untuk memainkannya adalah, bagaimana *Rhythm Game* bisa memiliki tingkat dari mudah dan susah namun tetap sesuai/pas dengan lagu.

2.2 Midi

Midi adalah sebuah standar internasional yang digunakan untuk bertukar data antar *hardware* dan *software*, pertukaran tersebut berupa kode musik atau *midi event*. *Midi event* akan berisi tiga elemen yaitu *channel*, *note*, *velocity*. *Channel* akan berisi suatu perintah singkat berupa jenis aktivitas apa yang akan dilakukan dalam midi, apakah membunyikan suatu suara atau menghentikan suatu suara selain itu juga *channel* akan merepresentasikan apa aktivitas yang akan dilakukan dalam midi, aktivitas tersebut bisa berupa *noteOn* yaitu aktivitas dimana note akan mulai dibunyikan dan *noteOff* yaitu aktivitas dimana note akan mulai dihentikan, selain aktivitas tersebut, *channel* juga akan merepresentasikan pada instrumen keberapa aktivitas tersebut dilakukan (ada 16 instrumen pada *midi* secara *general*), *midi channel* dalam kepentingan pemrograman akan memiliki range dari 0x80 hingga 0x8F untuk

kepentingan *noteOff* dari instrumen 1-16 dan 0x90 hingga 0x9F untuk kepentingan *noteOn* dari instrumen 1-16. 16 instrumen tersebut akan dijabarkan pada **Tabel 2.1** berikut.

Tabel 2.1 Tabel Instrumen yang mewakili tiap *channel* pada midi secara umum .

Channel	Instrument	*Channel*	Intrument
1	Grand Piano Akustis	9	*Celesta*
2	Piano Akustis	10	*Glockenspiel*
3	Grand Piano Elektrik	11	*Music Box*
4	Piano *Honky Tonk*	12	*Vibraphone*
5	Piano Elektrik 1	13	*Marimba*
6	Piano Elektrik 2	14	Xylophone
7	Piano Kuno (*Harpsichord)*	15	*Tubular Bells*
8	Klavinet	16	*Dulcimer*

Note akan berisi suatu perintah singkat berupa nada keberapa suatu instrumen yang didefinisikan oleh *channel* tersebut akan dibunyikan, *note* akan memiliki *range* 0 hingga 127. *Velocity* akan berisi perintah yang berupa kerasnya suatu suara semakin besar *velocity* semakin besar pula suara yang akan dibunyikan oleh *channel* selain itu juga beberapa midi, komposer memanfaatkan *velocity = 0* sebagai *noteOff* pada *channel* apabila *channel* tersebut tidak bisa melakukan *noteOff* dengan sendirinya. *Range* pada *velocity* juga sama dengan *note* yaitu 0 hingga 127.

2.3 Beats Per Minute (BPM)

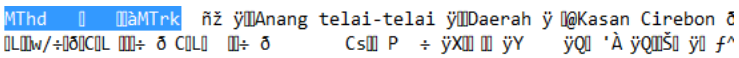
BPM atau *Beats Per Minute* adalah suatu satuan yang paling biasa digunakan sebagai pernyataan besarnya tempo lagu dan detak jantung. BPM menyatakan dalam setiap satu menitnya ada berapa ketukan/*beat*. Dengan alat ukur metronom beat bisa berupa ¼ ketukan, ½, ¾ serta ketukan penuh 4/4.

2.4 Midi File Handler

Midi File Handler adalah sebuah fungsi yang dijalankan oleh komputer, untuk memproses file midi menjadi informasi-informasi tertentu. Informasi utama yang didapatkan oleh Midi File Handler adalah *midi event*. Proses agar Midi File Handler dapat menyusun *midi event* adalah sebagai berikut:

1. Midi File Handler akan menerima file midi. sebelum itu file midi tersebut harus berekstensi .bytes, supaya Midi File Handler menerima file midi tersebut. Ketika file midi berhasil diterima maka Midi File Handler akan mulai membaca *chunk* dalam file midi tersebut, dimana *chunk* adalah bagian-bagian dari struktur file midi.
2. Setelah itu sebelum Midi File Handler melakukan pembacaan terhadap file midi, Midi File Handler memastikan apabila file midi tersebut memiliki struktur *chunk header* sebagai *chunk* awal dari file midi, *chunk header* dalam file midi akan diwakilkan dengan adanya penulisan "MThd" di awal file midi lalu dilanjutkan dengan mengetahui format, jumlah *track chunk* serta *division* (jarak yang memisahkan antar track), ketiga hal tersebut akan direpresentasikan dalam bentuk bytes dan harus sepanjang 6 bytes (format sepanjang 2 bytes, jumlah *track chunk* sepanjang 2 bytes dan *division* 2 bytes) apabila panjang *header chunk* tersebut tidak sama dengan 6 bytes, maka file midi akan bersifat *invalid*.
3. Apabila "MThd" sudah diketahui oleh Midi File Handler dan panjang *header chunk* sudah sesuai, maka Midi File Handler akan mulai membaca *chunk* selanjutnya yaitu pada bagian track *chunk*, *track chunk* harus diawali dengan adanya penulisan "MTrk" apabila tidak, file midi tersebut tidak dapat dilanjutkan pembacaanya.

Potongan file midi dengan tipe *chunk* yang benar dapat dilihat pada **Gambar 2.1**.



Gambar 2.1 Potongan File Midi dengan tipe *Chunk* yang tepat

4. Apabila pada *track chunk* terdapat "MTrk" maka *track chunk* sudah bersifat valid, sehingga Midi File Loader dapat melanjutkan pembacaan *track chunk* tersebut sehingga menghasilkan sebuah *midi event*. *Midi event* akan disimpan kedalam class, informasi waktu untuk *midi event* tersebut juga akan disimpan.
5. Setelah itu Midi File Handler juga memiliki *track sequencer* untuk membaca data *track* yang telah disimpan sebelumnya terutama *midi event*. Fungsi *track sequencer* yang memanggil *midi event* tersebut akan berjalan dengan kecepatan sesuai dengan BPM yang ditentukan. Dari sinilah *midi event* ini akan digunakan untuk menjadi perintah kepada *hardware* untuk melakukan sesuatu pada *midi event* tersebut (termasuk dengan membunyikan suara sesuai dengan ketentuan dengan *midi event* yang secara sekuensial diperoleh).

2.5 SMFLite

SMFLite adalah *class library* yang di buat oleh Keijiro Takahashi yang berfungsi sebagai Midi File Handler khusus *Standard Midi File*. SMFLite bersifat *open source* dan telah dibagikan melalui Github.com. SMFLite memiliki fungsi dalam membaca *midi event* tanpa melakukan *playback* pada file midi. Sehingga SMFLite dapat berjalan dengan sangat ringan. SMFLite akan berjalan dengan cara mengubah file midi menjadi file dengan ekstensi .bytes sehingga dapat dibaca oleh SMFLite, lalu apabila program dijalankan SMFLite akan melakukan *track reading* untuk

menyusun *midi event* yang akhirnya digunakan untuk *midi sequencing* dengan BPM yang telah ditentukan agar output akhirnya dihasilkan, dimana dalam aktivitas tersebut akan menghasilkan *midi event*. SMFLite adalah *plugin* yang sederhana yang tepat digunakan untuk visualisasi terhadap midi serta pemanfaatan *Rhythm Games*.

2.6 Unity

Unity adalah sebuah *game engine* yang berbasis *cross-platform*. Unity dapat digunakan untuk membuat sebuah game yang bisa digunakan pada perangkat komputer, ponsel pintar android, iPhone, PS3 dan bahkan X-BOX.

Unity adalah sebuah sebuah alat yang terintegrasi untuk membuat game, arsitektur bangunan dan simulasi. Unity dapat digunakan untuk membangun games PC dan games Online.

Fitur *scripting* yang disediakan, mendukung 3 bahasa pemrograman, JavaScript, C# dan Boo. *Moving, rotating* dan *scaling* objek hanya perlu sebaris kode. Begitu juga dengan *duplicating, removing* dan *changing properties*. *Visual Properties Variables* yang didefinisikan dengan *scripts* ditampilkan pada editor. Bisa digeser melalui *drag and drop* dan bisa memilih warna dengan *color picker*.

(Halaman Ini Sengaja dikosongkan)

BAB III
ANALISIS DAN PERANCANGAN

Bab ini akan membahas mengenai analisis dan perancangan sistem game *"Indonesian's Rhythm Game"*. Secara garis besar akan membahas tentang bagaimana sistem dasar untuk permainan agar sebisa mungkin bersifat otomatis, bagaimana UI yang terbentuk untuk aplikasi ini serta skenario antara pemain (pengguna) dengan aplikasi tersebut.

3.1 Perancangan Permainan

3.1.1 Deskripsi Umum Perangkat Lunak

Aplikasi yang akan dibuat adalah sebuah *Rhythm Game* dengan sistem *scrolling* 2D (dari atas ke bawah) yang akan dibuat untuk Windows PC dengan menggunakan input dari mouse untuk interaksi menu dan keyboard untuk melakukan interaksi saat game dimulai nantinya. *Rhythm Game* ini akan bertemakan tradisional Indonesia sehingga koleksi musiknya adalah lagu daerah dalam bentuk midi oleh karena itu game ini nantinya akan berjudul "*Indonesian's Rhythm Game"*.

Pengguna utama dari permainan ini adalah pengguna dengan hampir semua umur namun setidaknya mengerti sistem *Rhythm Game*. Pemain dapat memilih lagu yang sudah di sediakan pada aplikasi yang memiliki kesusahan yang berbeda-beda, dengan satuan level *range* yang paling mudah 1 hingga yang paling susah 5. Tingkat level akan berbeda di setiap lagu, setiap lagu juga akan berbeda-beda pula *channel* yang akan menjadi *note* dalam permainan.

Dalam aplikasi ini akan dibangun sebagai bentuk antisipasi terhadap kesusahan dalam membuat *note map* yang menyusun *note* dengan pas dan sesuai dengan lagu. Dengan memanfaatkan Midi File Handler, *note map* akan dibuat secara dinamis dengan waktu bersamaan saat suatu lagu dalam *game* dimainkan oleh pemain/pengguna. Antara *note map* dengan audio lagu akan diberi

waktu *delay* agar lagu dapat mulai berbunyi dengan *timing* yang pas dengan *note* yang akan menuju aktivator, sehingga nantinya pemain bisa mendapatkan akurasi sebaik mungkin karena dapat mengikuti *note* dengan irama lagu. Apabila *note designer* masih tetap ada campur tangan, *note designer* tidak perlu terlalu susah untuk membuat *note* agar memiliki *timing* yang pas, *note designer* hanya perlu menentukan waktu *delay* agar *note* yang dibuat sesuai dengan audio yang dibunyikan.

Aplikasi ini akan dibuat dengan aplikasi IDE Unity Editor 2017.2.2f1 dengan bahasa pemrograman C# khusus Unity. Aplikasi ini akan menggunakan Midi File Handler *library* "*SMFLite"* yang menggunakan bahasa C# sebagai *plugin*, ini akan menjadi komponen penting pada aplikasi ini untuk membaca file midi dalam bentuk file ekstensi .bytes, agar menghasilkan suatu informasi berupa *midi event*, yang nantinya digunakan untuk pembuatan *note* secara dinamis. Selain itu design pada *asset* yang akan digunakan pada game, sebagian besar akan dibuat dengan Adobe Photoshop CC 2015.

3.1.2 Spesifikasi Kebutuhan Fungsional

Berdasarkan deskripsi umum sistem, maka disimpulkan bahwa kebutuhan fungsional dari aplikasi ini hanya ada satu yaitu bermain game.

3.1.3 Spesifikasi Kebutuhan Non-Fungsional

Terdapat beberapa kebutuhan non-fungsional yang apabila dipenuhi, dapat meningkatkan kualitas dari permainan ini. Berikut daftar kebutuhan non-fungsional:

1. *FrameRate*

Permainan *Rhythm Game* sangat menitikberatkan kelancaran pada tampilan karena pada *Rhythm Game* pemain harus konsentrasi agar dapat mencapai ketepatan yang setepat-tepatnya, oleh karena itu *game* tersebut harus memiliki tampilan yang lancar, tidak ada *lag*

maupun *frame skipping*. Apabila ada *lag* atau *frame skipping* kemungkinan besar akan mempengaruhi kinerja pemain sehingga interaksi antara pemain dengan aplikasinya menjadi tidak sesuai. Standarnya *Rhythm Game* 2D hanya perlu berjalan dengan frame 30 FPS namun harus stabil (tidak meningkat drastis maupun menurun drastis).

2. Kebutuhan Grafis

Dalam *Rhythm Game*, juga perlu adanya suatu efek secara grafis agar dapat membedakan kondisi. Seperti apabila *note* sudah mencapai aktivator dan berhasil ditangkap maka aktivator akan *zoom up* sebagai tanda bahwa pemain telah berhasil menangkap *note* tersebut atau tidak. Dengan adanya tambahan efek grafis, secara tidak langsung dapat mempengaruhi pemahaman pemain terhadap game.

3.1.4 Karakteristik Pengguna

Berdasarkan deskripsi umum di atas, maka dapat diketahui bahwa pengguna yang akan menggunakan aplikasi ini ada dua orang yaitu, pemain yang memainkan permainan dan pengembang. Karakteristik pengguna tercantum dalam **Tabel 3.1**.

Tabel 3.1 Karakteristik Pengguna

Nama Aktor	Tugas	Hak Akses Aplikasi	Kemampuan yang harus dimiliki
Pemain	Pihak luar yang memainkan permainan.	Memainkan permainan.	Tidak ada.

3.2 Perancangan Sistem

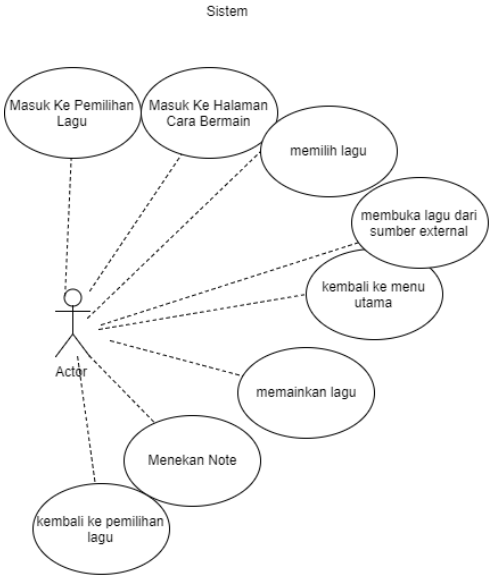
Tahap perancangan dalam subbab ini dibagi menjadi beberapa bagian yaitu perancangan diagram kasus penggunaan, perancangan skenario kasus penggunaan, perancangan antarmuka,

perancangan aturan permainan, perancangan data. Dengan ini sistem yang akan dibangun akan dijelaskan dengan lengkap.

3.2.1 Perancangan Diagram Kasus Penggunaan

Dalam aplikasi tugas akhir ini, terdapat tujuh kasus penggunaan yaitu diantaranya adalah masuk ke pemilihan lagu, masuk ke halaman cara bermain, memilih lagu, kembali ke menu utama, memainkan lagu, menekan *note* dan kembali ke pemilihan lagu. Aktor dari kasus penggunaan adalah pemain.

Kasus penggunaan yang terdapat didalam sistem dicantumkan pada **Gambar 3.1**.



Gambar 3.1 Diagram kasus aplikasi

3.2.2 Perancangan Skenario Kasus Penggunaan

Penjelasan dari masing-masing kasus penggunaan dicantumkan pada **Tabel 3.2**. Tabel tersebut berisi penjelasan skenario yang akan dilakukan ketika pengujian.

Tabel 3.2 Skenario Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-001	Masuk ke Pemilihan Lagu	Untuk menampilkan halaman pemilihan lagu.
2	UC-002	Masuk ke Halaman Cara Bermain	Untuk menampilkan halaman cara bermain.
3	UC-003	Memilih Lagu	Untuk memilih lagu, lalu memunculkan tampilan tentang lagu yang dipilih.
4	UC-004	Membuka Lagu dari File Eksternal	Untuk memilih lagu yang dimiliki sendiri dengan memasukan file .wav (karena file .mp3 tidak didukung), file .bytes dan file .ini dari folder diluar folder project.
5	UC-005	Kembali ke Menu Utama	Untuk mengembalikan ke tampilan menu utama.
6	UC-006	Memainkan Lagu	Memainkan lagu hingga selesai.
7	UC-007	Menekan *Note*	Menekan *note* apabila *note* sudah sampai / mendekati aktivator.
8	UC-008	Kembali ke Pemilihan Lagu	Kembali ke halaman pemilihan lagu apabila lagu sudah selesai baik menang maupun kalah.

3.2.2.1 Kasus Penggunaan Permainan

Penjelasan kasus penggunaan permainan untuk skenario UC-001 yaitu Masuk ke Pemilihan Lagu dijelaskan pada **Tabel 3.3**.

Tabel 3.3 Skenario Kasus Penggunaan Masuk ke Pemilihan Lagu

Nama Kasus Penggunaan	Masuk ke Pemilihan Lagu
Kode	UC-001
Deskripsi	Kasus penggunaan dimana aktor akan memilih menu mulai bermain sehingga daftar lagu di tampilkan.
Aktor	Pemain
Kondisi Awal	Pemain sudah masuk ke aplikasi dan sistem menampilkan tampilan main menu.
Alur Normal	1. Pemain menekan tombol “mulai bermain”. 2. Sistem menampilkan daftar lagu.

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-002 yakni Masuk ke Halaman Cara Bermain dijelaskan pada **Tabel 3.4**.

Tabel 3.4 Skenario Kasus Penggunaan Masuk ke Halaman Cara Bermain

Nama Kasus Penggunaan	Masuk ke Halaman Cara Bermain
Kode	UC-002
Deskripsi	Kasus penggunaan dimana aktor bermain akan memilih menu cara bermain agar menuju halaman cara bermain.
Aktor	Pemain
Kondisi Awal	Pemain sudah masuk ke aplikasi dan sistem menampilkan tampilan main menu.

Alr Normal	1. Pemain mengklik tombol “mulai bermain”. 2. Sistem menampilkan halaman cara bermain. 3. Sistem memulai video *tutorial*.

Kemudian penjelasan kasus penggunaan permainan untuk skenario UC-003 yakni Memilih Lagu dijelaskan pada **Tabel 3.5**.

Tabel 3.5 Skenario Kasus Penggunaan Memilih Lagu

Nama Kasus Penggunaan	Memilih Lagu
Kode	UC-003
Deskripsi	Kasus penggunaan dimana aktor akan Memilih Lagu sehingga detail lagu di munculkan.
Aktor	Pemain
Kondisi Awal	Pemain sudah masuk ke halaman pemilihan lagu.
Alur Normal	1. Pemain mengeklik salah satu lagu yang diinginkan. 2. Sistem menunjukkan detail lagu, dari nama, BPM, serta level. 3. Sistem menampilkan Tombol ‘Mulai’.

Kemudian penjelasan kasus penggunaan permainan untuk skenario UC-004 yakni Membuka Lagu dari sumber Eksternal, dijelaskan pada **Tabel 3.6**.

Tabel 3.6 Skenario Kasus Membuka Lagu dari sumber Eksternal

Nama Kasus Penggunaan	Membuka Lagu dari sumber Eksternal
Kode	UC-004
Deskripsi	Kasus penggunaan dimana aktor akan membuka file .bytes, file .wav (karena mp3 eksternal tidak didukung Unity) dan file .ini milik sendiri untuk dimainkan ke apilikasi Game.
Aktor	Pemain
Kondisi Awal	Sistem telah menampilkan halaman pemilihan lagu.

Alur Normal	1	Pemain memilih tombol ‘kustomisasi lagu’.
	2	Sistem menampilkan halaman kustomisasi lagu.
	3	Pemain klik membuka file .wav.
	4	Sistem menampilkan File Browser untuk mencari file .wav.
	5	Pemain klik open.
	6	Sistem mengambil file .wav tersebut dari folder eksternal.
	7	Pemain klik close.
	8	Sistem menampilkan nama lagu.
	9	Pemain klik membuka file .bytes.
	10	Sistem menampilkan File Browser untuk mencari file .bytes.
	11	Pemain klik open.
	12	Sistem mengambil file .bytes dari folder eksternal
	13	Pemain klik close.
	14	Pemain klik membuka file .ini.
	15	Sistem menampilkan File Browser untuk mencari file ini.
	16	Pemain klik open.
	17	Sistem mengambil file .ini tersebut dari folder eksternal.
	18	Pemain klik close.
	19	Sistem menampilkan BPM serta tingkat kesulitan pada lagu.
	20	Sistem menampilkan tombol ‘Mulai’.

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-005 yakni Kembali ke Manu Utama dijelaskan pada **Tabel 3.7**.

Tabel 3.7 Skenario Kasus Kembali ke Menu Utama

Nama Kasus Penggunaan	Kembali ke Menu Utama
Kode	UC-005

Deskripsi	Kasus penggunaan dimana aktor akan kembali ke menu utama.
Aktor	Pemain
Kondisi Awal	Sistem menampilkan tombol “Kembali”. Tombol tersebut terdapat pada halaman Pemilihan lagu dan cara bermain.
Alur Normal	1 Pemain memilih tombol ‘Kembali’. 2 Sistem kembali ke Menu Utama.

Kemudian penjelasan kasus penggunaan permainan untuk skenario UC-006 yakni Memainkan Lagu dijelaskan pada **Tabel 3.8**.

Tabel 3.8 Skenario Kasus Penggunaan Memainkan Lagu

Nama Kasus Penggunaan	Memainkan Lagu
Kode	UC-006
Deskripsi	Kasus penggunaan dimana aktor akan memainkan lagu yang telah dipilih.
Aktor	Pemain
Kondisi Awal	Pemain sudah memilih lagu atau membuka file lagu milik sendiri dan sistem sudah menampilkan yombol mulai.
Alur Normal	1 Pemain menekan tombol mulai. 2 Sistem menampilkan halaman permainan yang segera dimulai. 3 Sistem akan memunculkan *note* selama lagu berlangsung . 4 Pemain menekan *Note* yang akan datang setepat mungkin selama lagu berlangsung (UC-006). 5 Alur pada nomor 3-4 akan berulang hingga apabila lagu selesai dengan *life gauge* bersisa hingga sistem menampilkan halaman “Result” sedangkan apabila *life gauge* habis sebelum lagu selesai, sistem akan menampilkan halaman gagal/”sayang sekali”.

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-007 yakni Menekan *Note* dijelaskan pada **Tabel 3.9**.

Tabel 3.9 Skenario Kasus Penggunaan Menekan *Note*

Nama Kasus Penggunaan	Menekan *Note*
Kode	UC-007
Deskripsi	Kasus penggunaan dimana Pemain menekan *note* saat *note* sudah menuju aktivator.
Aktor	Pemain
Kondisi Awal	Sistem menampilkan tulisan level complete pada suatu level.
Alur Normal	1. Pemain sebisa mungkin menekan tombol input apabila *note* sudah mencapai aktivator. 2. Sistem akan memutuskan apakah *note* berhasil ditangkap oleh aktivator atau tidak. 3. Apabila berhasil maka, sistem akan menaikan *life gauge*, menambah *combo*/rantai dan memutuskan *judgement* apa yang didapat, jika gagal maka *judgement* akan langsung menjadi "buruk" dan me-*reset* combo/rantai menjadi 0 dan *life gauge berkurang*. 4. Apabila *judgement* yang didapat "sempurna", maka skor akan ditambah 100, sedangkan "mantap" akan menambah skor sebanyak 75 dan "ya sudahlah " akan menambah skor sebanyak 50.

Selanjutnya penjelasan kasus penggunaan permainan untuk skenario UC-008 yakni Kembali ke Pemilihan Lagu dijelaskan pada **Tabel 3.10**.

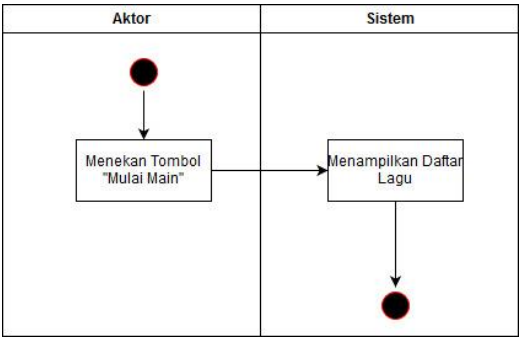
Tabel 3.10 Skenario Kasus Penggunaan Kembali ke Pemilihan Lagu

Nama Kasus Penggunaan	Kembali ke Pemilihan Lagu
Kode	UC-008
Deskripsi	Kasus penggunaan dimana pemain akan kembali ke pemilihan lagu untuk memilih lagu selanjutnya.
Aktor	Pemain
Kondisi Awal	Sistem menandakan bahwa sebuah lagu sudah selesai dengan menampilkan halaman "Result" apabila menang atau "gagal" apabila kalah.
Alur Normal	1. Pemain menekan tombol "selanjutnya". 2. Sistem akan menampilkan kembali Pemilihan Lagu.

3.2.2.2 Diagram Aktivitas

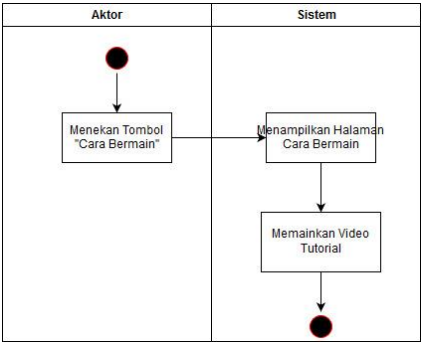
Diagram Aktivitas menampilkan langkah-langkah normal yang harus dilakukan pemain untuk menjalankan studi kasus permainan dimulai dari awal permainan hingga kondisi akhir.

Diagram Aktivitas untuk dari kasus penggunaan UC-001 yakni Masuk ke Pemilihan Lagu dijelaskan pada **Gambar 3.2**.



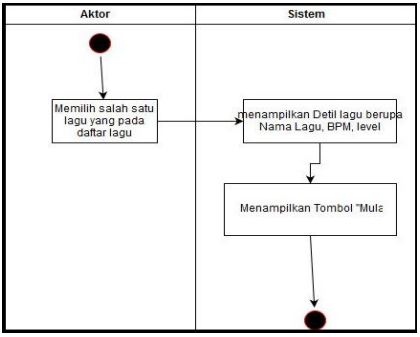
Gambar 3.2 Diagram Aktivitas Masuk ke Pemilihan Lagu

Kemudian Diagram Aktivitas untuk dari kasus penggunaan UC-002 yakni Masuk ke Halaman Cara Bermain dijelaskan pada **Gambar 3.3**. pada halaman Cara Bermain, terdapat sebuah video yang menunjukkan cara bermain pada *Rhythm Game* ini.



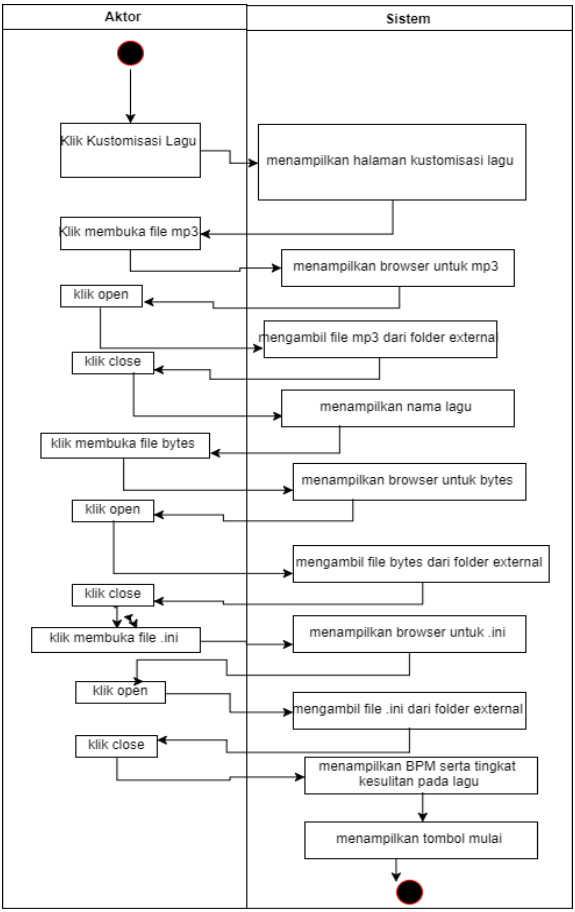
Gambar 3.3 Diagram Aktivitas Masuk ke Halaman Cara Bermain

Kemudian Diagram Aktivitas untuk dari kasus penggunaan UC-003 yakni Memilih Lagu pada **Gambar 3.4**. dimana pemain akan menekan salah satu daftar lagu yang berupa tombol lalu sistem akan menampilkan detail lagu yang dipilih tersebut.



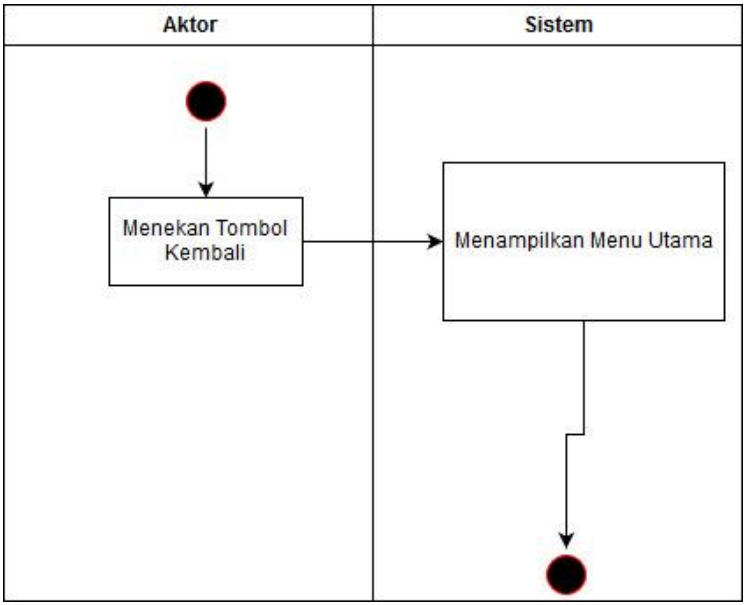
Gambar 3.4 Diagram Aktivitas Memilih Lagu

Kemudian Diagram Aktivitas untuk dari kasus penggunaan UC-004 Menbuka Lagu dari file Eksternal dijelaskan pada pada **Gambar 3.5**. pemain akan klik kustomisasi lagu untuk kehalaman untuk membuka file, halaman tersebut dilengkapi dengan file browser.



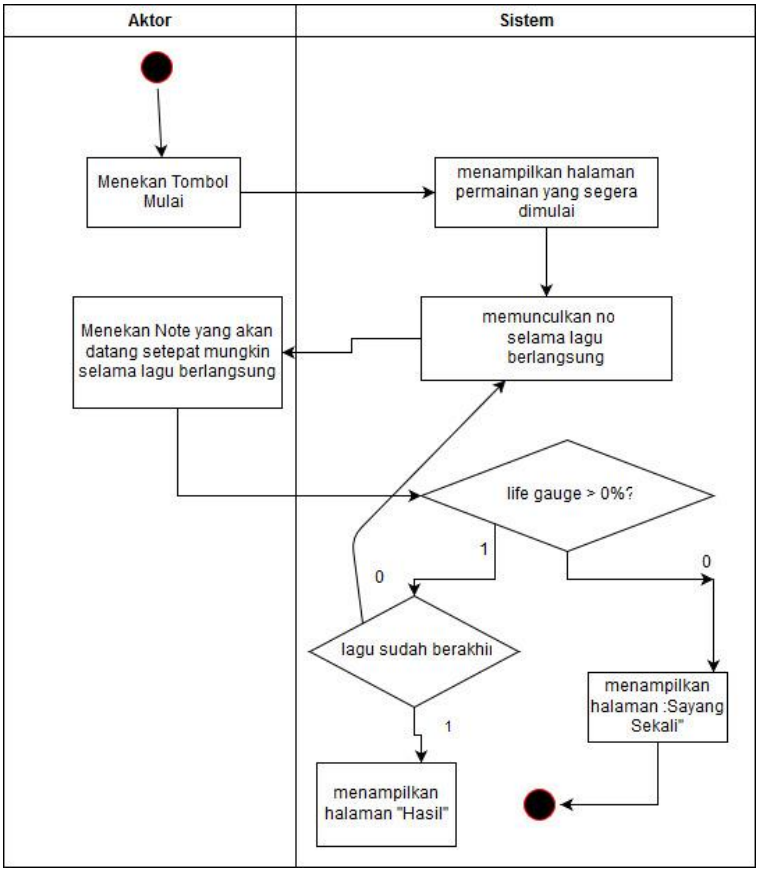
Gambar 3.5 Diagram Aktivitas Membuka File Lagu dari sumber eksternal

Kemudian Diagram Aktivitas untuk dari kasus penggunaan UC-005 yakni Kembali ke Menu Utama dijelaskan pada **Gambar 3.6**. Diagram Aktivitas menyatakan bahwa aktivitas dimulai dari kondisi awal setelah aktivitas dari kasus penggunaan UC-001 atau UC-002 telah dilakukan, sehingga terdapat tombol "kembali ".



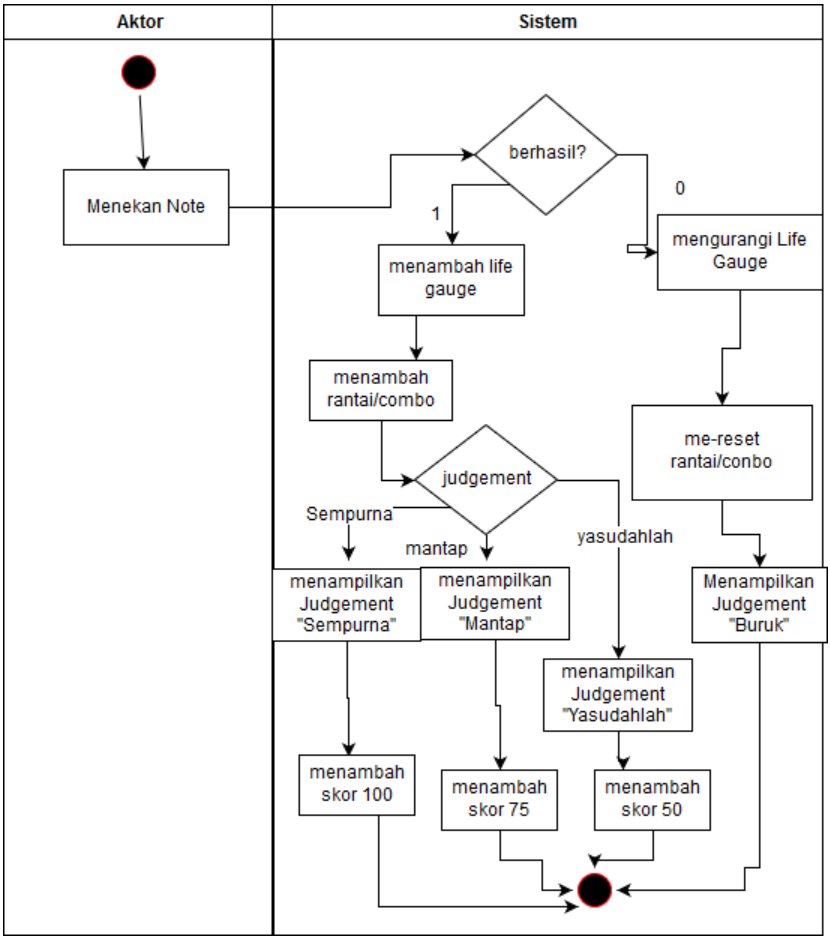
Gambar 3.6 Diagram Aktivitas Kembali ke Menu Utama

Selanjutnya Diagram Aktivitas untuk dari kasus penggunaan UC-006 yakni Memainkan Lagu dijelaskan pada **Gambar 3.7**. pada kasus penggunaan ini diawali dengan kondisi dimana lagu sudah dipilih sehingga tombol mulai sudah muncul. Pada kasus penggunaan ini Midi File Handler akan beskerja, selama lagu masih berjalan dan *life gauge* masih belum 0%, maka lagu akan terus berlangsung hingga selesai.



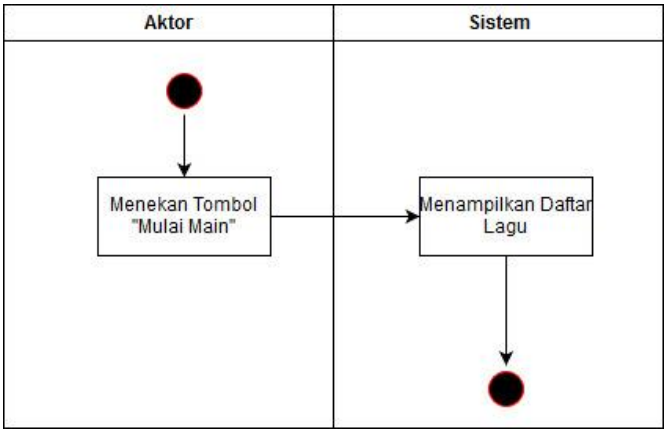
Gambar 3.7 Diagram Aktivitas Memainkan Lagu

Kemudian Diagram Aktivitas untuk dari kasus penggunaan UC-007 yakni Menekan *Note* yang merupakan kasus penggunaan yang melengkapi bagian dari UC-006 yaitu saat Menekan *Note* saat lagu berlangsung, akan dijelaskan pada **Gambar 3.8**. pada kasus penggunaan ini akan ada perkondisian apa yang akan terjadi apabila seorang pemain menekan *note* sehingga mempengaruhi skor, *combo*/rantai dan *life gauge*.



Gambar 3.8 Diagram Aktivitas Menekan *Note*

Kemudian Diagram Aktivitas untuk Penggunaan UC-008 dimana pemain akan Kembali ke Pemilihan Lagu setelah berada di halaman “gagal” atau “hasil” dapat dilihat pada **Gambar 3.9**.



Gambar 3.9 Diagram Aktivitas Kembali ke Pemilihan Lagu

3.2.3 Perancangan Antarmuka Pengguna

Subbab ini membahas bagaimana rancangan antarmuka pengguna yang akan digunakan untuk tugas akhir. Rancangan antarmuka yang dibahas meliputi ketentuan beberapa objek yang akan digunakan dan rancangan jendela tampilan. Dalam aplikasi ini terdapat beberapa tampilan, yaitu main menu, pemilihan lagu, cara bermain, main game, result/hasil, gagal/"sayang sekali".

3.2.3.1 Tampilan Main Menu

Tampilan Main Menu merupakan tampilan yang pertama kali muncul ketika aplikasi dijalankan untuk yang pertama kalinya. Tampilan ini terdiri dari judul serta 2 menu utama. Tampilan ini diilustrasikan pada **Gambar 3.10**.

3.2.3.2 Tampilan Pemilihan Lagu

Tampilan Pemilihan Lagu merupakan tampilan yang muncul setelah pengguna menekan tombol mulai main. Tampilan ini terdiri dari sebuah *list*/daftar yang berisi tombol yang diurut dari atas

kebawah, tombol yang diurutkan ini akan berisi nama lagu, selain itu terdapat sebuah *scrollbar* untuk menuju ke bawah list apabila lagunya banyak. Lalu ada satu tombol untuk kembali ke menu utama. Tampilan ini ditunjukkan pada **Gambar 3.11**.

Tampilan Pemilihan Lagu akan terjadi perubahan ketika salah satu button yang berisi lagu pada *list* telah ditekan, maka pada samping kanan *list* akan menampilkan nama lagu yang dipilih, BPM serta tingkat kesulitan/level pada lagu dan disamping kanan tombol kembali akan terdapat tombol mulai main yang akan menghantar Pemain ke tampilan main game nantinya. Tampilan ini diilustrasikan pada **Gambar 3.12**.

3.2.3.3 Tampilan Kustomisasi Musik

Tampilan ini adalah tampilan dimana pemain dapat membuka file .wav, file .bytes dan ini untuk dimainkan menuju game. Tampilan ini terdiri dari tulisan atas yang akan memberikan notifikasi tentang file yang telah dibuka, lalu tombol untuk open file .wav, file .bytes dan file .ini. Tampilan ini akan ditunjukkan pada **Gambar 3.13**. Lalu setelah klik salah satu dari tiga tombol pada sisi kiri bawah, maka akan tampil file browser yang seperti ditampilkan pada **Gambar 3.14**. Setelah semua file dimasukan maka akan muncul nama lagu dan tombol untuk memulai seperti yang diilustrasikan pada **Gambar 3.15**.

3.2.3.4 Tampilan Main Game

Tampilan ini merupakan tampilan inti dari game dan merupakan tanda bahwa game telah dimulai. Game ini akan menampilkan komponen yang cukup banyak yaitu, tampilan skor, rantai dan *judgement* dalam bentuk teks. Lalu pada bagian tengah adalah sebuah *layout game* yang dibawahnya terdapat aktivator A, S, D, J , K, L. Pada *layout* inilah *note* akan datang menuju aktivator dari atas ke bawah. Lalu pada samping kanan tengah, terdapat sebuah bar yang bergradasi, menunjukkan bahwa bar tersebut adalah

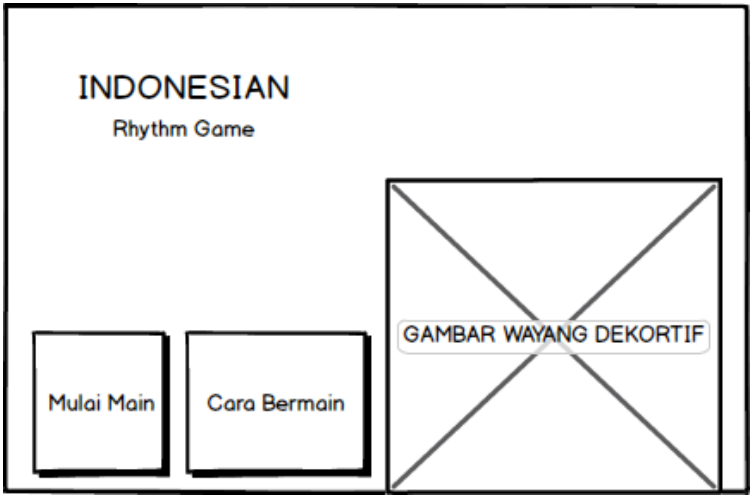
life gauge, yang akan menentukan menang kalahnya saat game berlangsung. Tampilan ini diilustrasikan pada **Gambar 3.16**.

3.2.3.5 Tampilan Hasil

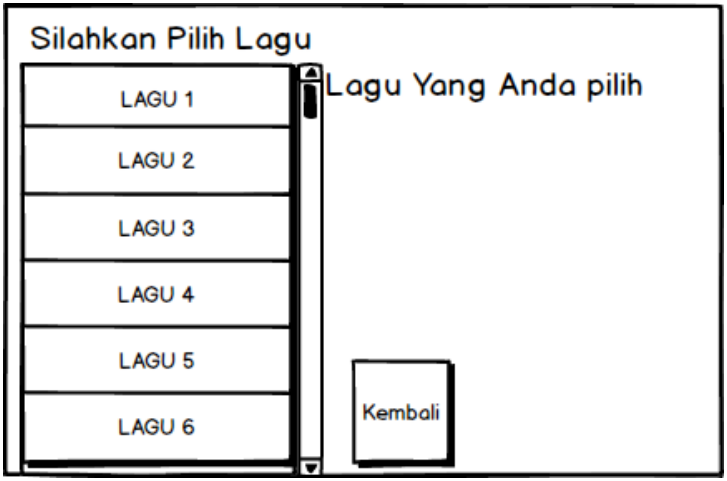
Tampilan ini merupakan tampilan yang menunjukkan bahwa Pemain berhasil bertahan hingga akhir dari sebuah lagu. Tampilan ini akan terdiri dari banyak teks yang menunjukkan sebuah hasil dari hasil main pada tampilan Main Game, teks tersebut akan terdiri dari jumlah skor/nilai, jumlah rantai/*combo* terpanjang, jumlah notasi yang lewat pada lagu yang telah dimainkan, lalu dibawahnya terdapat rekapitulasi *judgement*, yang menunjukkan banyaknya jumlah "sempurna", "mantap", "ya sudahlah" dan "buruk" lalu dibawahnya lagi adalah persentase keseluruhan penilaian performa pemain, lalu di sebelah kanannya terdapat sebuah tombol dengan tulisan "selanjutnya" untuk kembali ke halaman pemilihan lagu. Tampilan ini ditunjukkan seperti pada **Gambar 3.17**.

3.2.3.6 Tampilan Gagal/"Sayang Sekali"

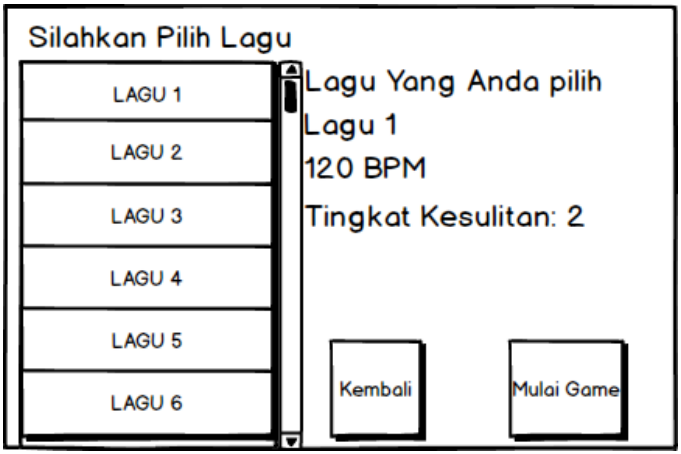
Tampilan ini merupakan tampilan dimana pemain dinyatakan tidak berhasil untuk bertahan hingga lagu berakhir , hal ini disebabkan oleh *needle* pada *life gauge* yang sudah mengarah ke pojok kiri pada *life gauge* yang menunjukkan bahwa *life gauge* telah habis. Tampilan ini hanya terdiri dari tulisan "wah sayang sekali coba lagi ya" lalu tombol dengan tulisan "selanjutnya" untuk menuju tampilan Pemilihan Lagu, hal ini sama dengan tombol "selanjutnya" yang terdapat di tampilan Hasil. Tampilan ini ditunjukkan seperti pada **Gambar 3.18**.



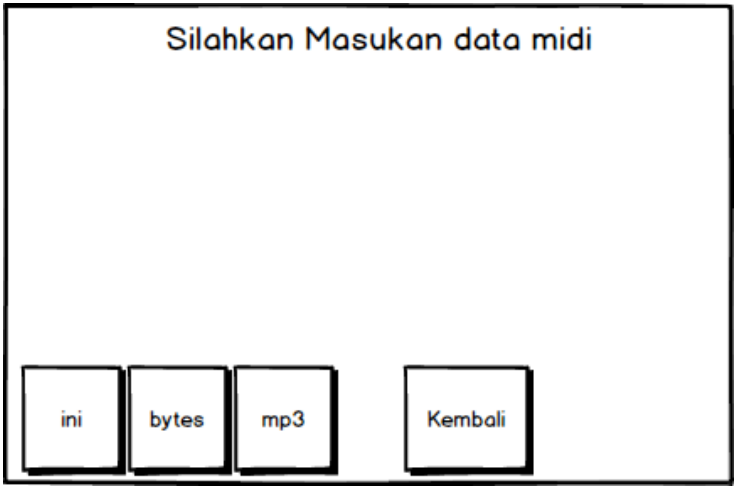
Gambar 3.10 Tampilan Main Menu



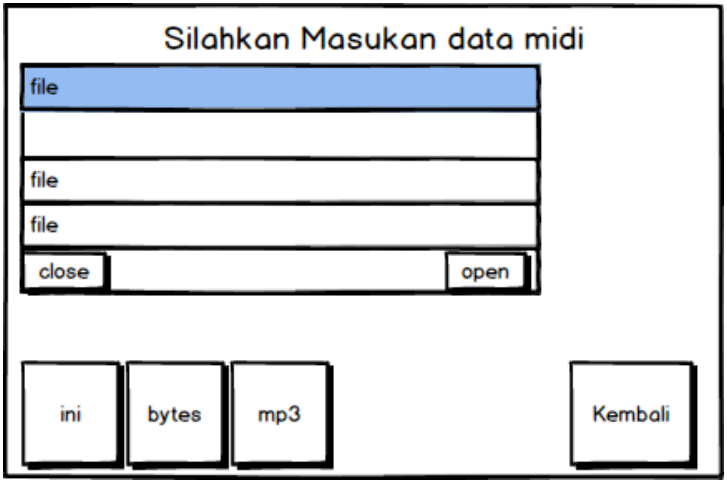
Gambar 3.11 Tampilan Pemilihan Lagu



Gambar 3.12 Tampilan Pemilihan Lagu setelah salah Satu Lagu dipilih

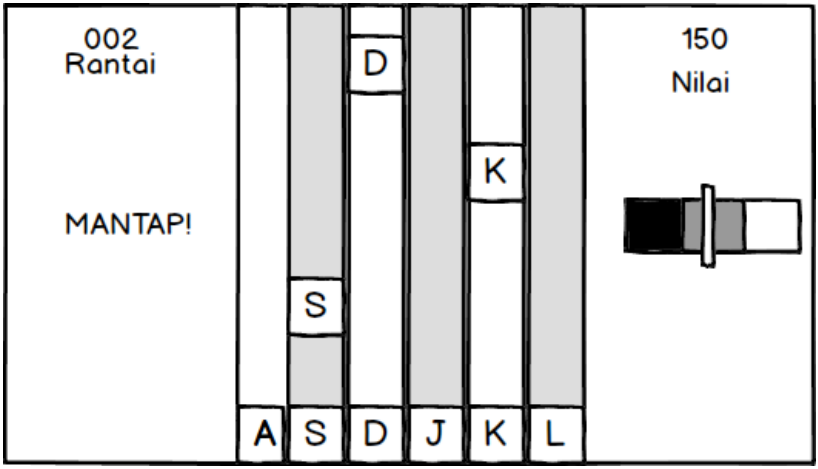


Gambar 3.13Tampilan Kustomisasi Musik



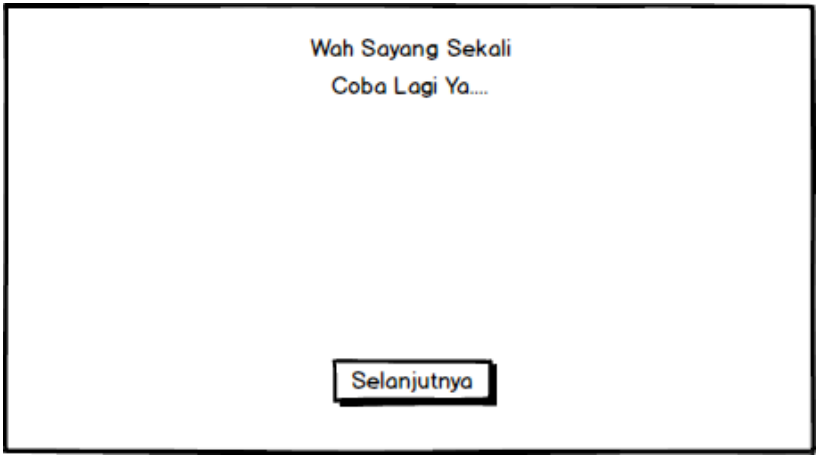
Gambar 3.14 File Browser pada halaman Kustomisasi

Gambar 3.15 Tampilan Kustomisasi Musik setelah data lengkap



Gambar 3.16 Tampilan Main game

Gambar 3.17 Tampilan Hasil



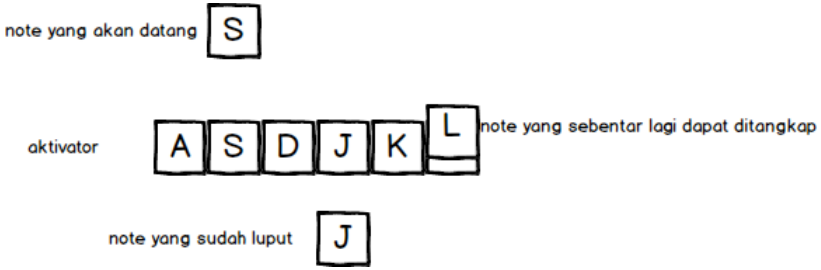
Gambar 3.18 Tampilan Gagal/"sayang sekali"

3.2.4 Perancangan Kontrol Permainan

Kontrol pada permainan ini ada 2 yaitu mouse yang digunakan untuk interaksi terhadap menu dan keyboard untuk permainan, input tersebut akan melibatkan 6 huruf yaitu A, S, D, J, K , L.

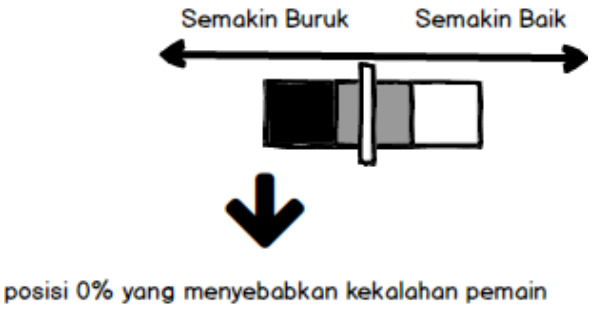
3.2.5 Perancangan Aturan Permainan

Dalam memainkan *Rhythm Game*, tujuan pemain adalah untuk menangkap *note* agar tidak lolos dari aktivator. Dalam menangkap *note* seorang pemain akan mendengarkan lagu yang diputar saat game dimulai, untuk membantu pemain dalam menangkap *note* dengan baik, illustrasi aktivator dengan *note* dapat dilihat pada **Gambar 3.19**.



Gambar 3.19 Illustrasi antara Aktivator, *Note* yang akan datang menuju Aktivator, *Note* yang sudah menyentuh Aktivator serta *Note* yang sudah lolos dari Aktivator

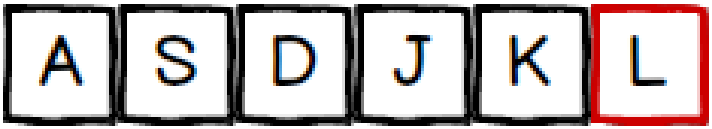
Apabila *note* lolos dari aktivator maka *life gauge* yang mewakili nyawa pemain akan berkurang dengan nilai tertentu, sedangkan apabila *note* berhasil ditangkap maka *life gauge* akan aman bahkan akan bertambah namun pada umumnya nilai tambah pada *life gauge* akan lebih sedikit daripada nilai berkurangnya *life gauge*, maka dari itu apabila *note* kebanyakan lolos dari aktivator maka akhirnya, game akan kalah karena pemain tidak dapat mempertahankan *life gauge* tersebut. Illustrasi *life gauge* ditunjukkan pada **Gambar 3.20**.



Gambar 3.20 Illustrasi *Life gauge*

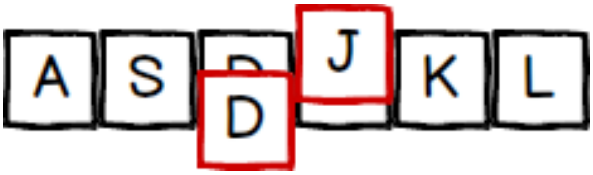
Selain *sistem life gauge*, dalam *Rhythm Game* biasanya akan memiliki sistem *judgement* apabila *note* sudah berhasil di tangkap oleh aktivator atau tidak, agar *Rhythm Game* bersifat tidak begitu mudah bagi para pemainnya dengan ini, skor tiap pemain akan variatif tergantung kemampuan dan akurasi pemainnya. Pada tugas akhir ini, *judgement* akan meliputi empat tingkat akurasi, yaitu kondisi "sempurna", "mantap" serta "yasudahlah" terjadi apabila *note* berhasil ditangkap dan kondisi "buruk" apabila note gagal ditekan.

Kondisi *judgement* pertama dan yang paling tinggi adalah kondisi "sempurna", ini adalah kondisi dimana *note* ditangkap disaat *note* benar-benar pas pada aktivator, kondisi ini akan memberikan penambahan skor sebanyak 100 dan persentase 100% pada satu *note*. Kondisi "sempurna" dapat dilihat pada **Gambar 3.21**, ditunjukkan bahwa kondisi "sempurna" tersebut adalah seperti *note* L yang berwarna merah yang benar-benar pas berada di posisi yang sama dengan aktivator L.



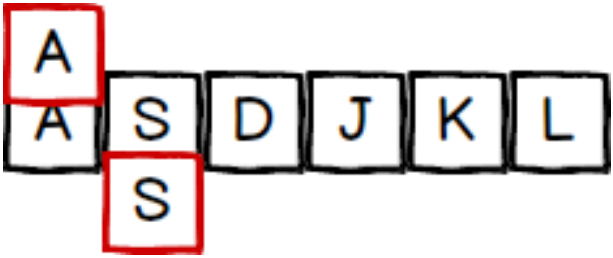
Gambar 3.21 Illustrasi Kondisi *Judgement* "sempurna"

Kondisi selanjutnya adalah kondisi "mantap", ini adalah kondisi dimana *note* tidak berada pada posisi yang benar-benar pas dengan aktivator, namun cukup dekat baik berada di atas aktivator maupun dibawah. Kondisi ini akan memberikan penambahan skor sebanyak 75 dari nilai sempurna dan persentase 75% pada satu *note*. Kondisi "mantap" diilustrasikan seperti pada **Gambar 3.22**. Dimana *note* J berada sedikit di atas dari aktivator J, namun masih dekat. Begitu juga dengan *note* D berada sedikit dibawah aktivator D namun masih dekat juga, ini akan menyebabkan kondisi "mantap".



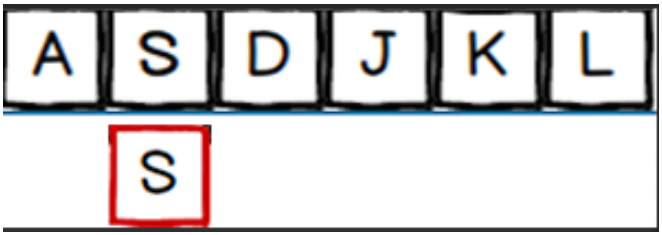
Gambar 3.22 Illustrasi Kondisi *Judgement* "mantap"

Kondisi terakhir pada game ini, yang termasuk kondisi dimana *note* berhasil ditangkap adalah kondisi "ya sudahlah". Kondisi ini adalah kondisi dimana *note* benar-benar tidak tepat pada aktivator namun masih menyentuh aktivator sehingga *note* masih berhasil ditangkap. Kondisi ini akan memberikan skor sebanyak 50 dan persentase 50% pada satu *note*. dimana *note* A berada di atas dari aktivator A dengan jarak yang cukup jauh namun masih menyentuh. Begitu juga dengan *note* S berada sedikit dibawah aktivator S yang hampir lolos namun masih menyentuh aktivator, ini akan menyebabkan kondisi "ya sudahlah". Kondisi "ya sudahlah" diilustrasikan seperti pada **Gambar 3.23**.



Gambar 3.23 Ilustrasi Kondisi *Judgement* "ya sudahlah"

Kondisi *judgement* terakhir dan yang paling bawah adalah kondisi "buruk" dimana *note* telah lolos/luput dari aktivator dan sama sekali tidak akan menambah skor, akurasi akan bernilai 0 %. Kondisi buruk dapat ditunjukkan pada **Gambar 3.24**.



Gambar 3.24 Ilustrasi Kondisi *Judgement* "buruk"

3.2.6 Perancangan Data

Perancangan data merupakan hal penting untuk diperhatikan karena diperlukan data yang tepat agar sistem beroperasi secara benar. Perancangan data ini dibagi menjadi dua yakni perancangan data midi untuk sistem dan data *note* untuk berlangsungnya permainan.

3.2.6.1 Perancangan File Midi dan konfigurasinya

Perancangan File Midi diawali dengan melakukan tes satu persatu *file midi* yang ada. Agar satu set lagu midi dapat dimainkan maka setiap file midi perlu adanya 3 file yaitu file .bytes, .mp3/.wav serta .ini dimana akan dijabarkan sebagai berikut.

- File .bytes
  File .bytes diperoleh dari file .mid yang hanya diubah ekstensinya menjadi .bytes. perubahan ekstensi ini berfungsi agar file midi dapat dibaca oleh Midi File Handler, selain itu pada file .bytes tersebut akan mengekstraksi data byte. Hasil pembacaan file .bytes inilah yang akan menghasilkan suatu *midi event* yang nantinya akan digunakan oleh *track sequencer* untuk menentukan posisi *note* pada waktu yang sesuai dengan kapan dipanggilnya *midi event* tersebut.

- File .mp3/.wav
  File .mp3/.wav adalah file yang digunakan sebagai media suara pada permainan. Unity tidak mendukung file .mid untuk langsung dimainkan melalui komponen *audio source*, Unity hanya mendukung file .ogg, file .mp3, file .wav dan file audio yang sudah bersifat fix lainya. Oleh karena itu file .mid harus dikonversi juga menjadi file .mp3 atau file .wav untuk dapat dimainkan dalam *audio source* Unity. Namun file file .mp3 hanya dapat dimainkan apabila file .mp3 tersebut bersifat internal atau telah dimasukan ke dalam folder project dan diproses oleh Unity, file .mp3 sebagai data eksternal tidak akan didukung oleh Unity karena disebabkan oleh versi Unity terbaru sudah tidak mendukung *UnityWebRequest* terhadap file mp3 akibat dari putusnya lisensi Unity dengan penemu mp3 tersebut. Karena permainan ini memiliki fitur untuk mengambil dapat mengambil file eksternal, maka untuk suara akan mengambil file dengan ekstensi .wav karena masih didukung oleh *UnityWebRequest*.

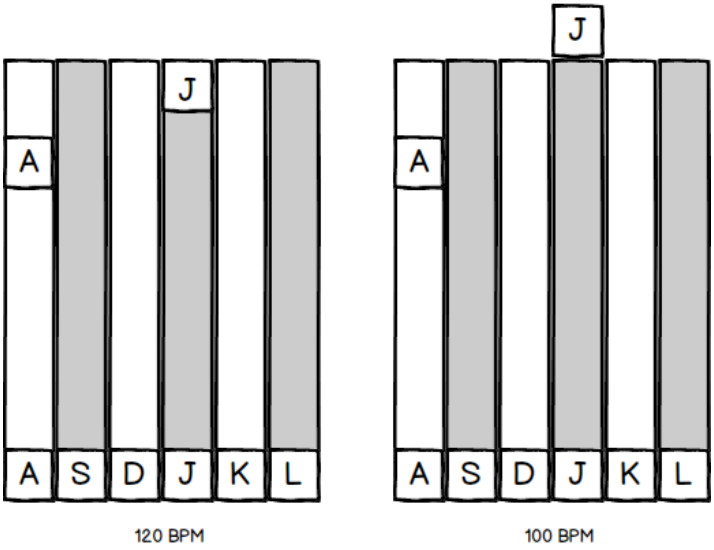
- File .ini
  File .ini adalah file konfigurasi yang dibuat secara manual untuk menyesuaikan file midi dalam permainan, dengan file .ini maka Midi File Handler akan mengetahui kapan *note* dibuat, berapa jarak waktu antara file .mp3 dengan file .bytes dan sebagainya.

File .ini yang sudah dibahas pada bagian sebelumnya, sangat penting agar lagu dalam permainan dapat bersifat teratur dan memperoleh *timing* yang tepat dan tingkat kesulitan yang sesuai dengan lagu, ada 8 hal yang akan dikonfigurasi dalam file .ini adalah sebagai berikut.

- **BPM**

Pertama-tama hal yang harus dikonfigurasi adalah BPM atau *Beats Per Minute* pada lagu midi tersebut. *Beats Per Minute* adalah satuan kecepatan yang digunakan pada lagu untuk menentukan

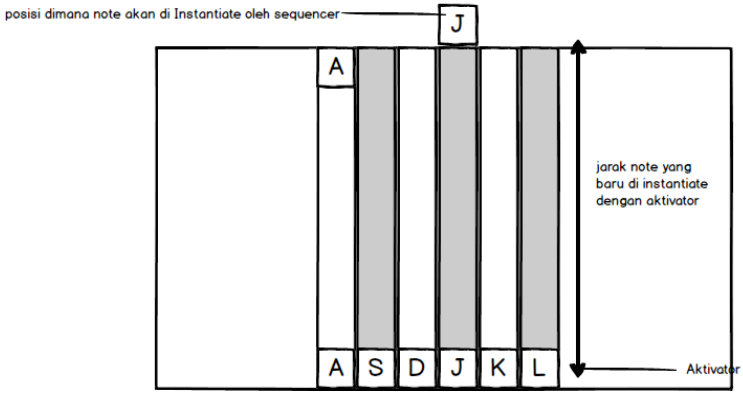
seberapa banyak ketukan/detakan pada suatu lagu. *Beats Per Minute*/BPM pada midi bisa dideteksi menggunakan aplikasi *Fruity Loops* (dianjurkan) dan aplikasi editor midi lainnya. Konfigurasi BPM pada *Rhythm Game* ini akan menjadi salah satu faktor yang menentukan kapan suatu *note* akan diinstansiasi. Semakin besar satuan BPM, maka semakin cepat juga *midi event* akan dipanggil sehingga semakin cepat juga *note* akan segera diinstansiasi. Apabila BPM tidak dikonfigurasi dengan tepat maka *note* yang akan terbentuk tidak akan pas dengan audio yang dibunyikan, lebih jelasnya, bahwa *note* akan datang ke aktivator pada irama yang tidak tepat dengan lagunya. Ilustrasi contoh perbandingan BPM dapat dilihat pada **Gambar 3.25**. Besar BPM akan dalam besaran *float*.



Gambar 3.25 Ilustrasi Perbedaan *Note* Map pada Lagu yang sama, waktu yang sama namun BPM berbeda

- **Waktu *Delay*/Tunda**

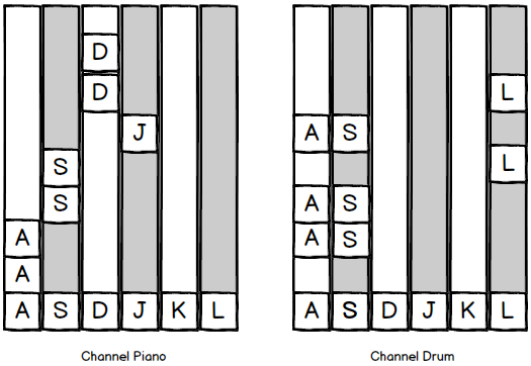
Lalu hal selanjutnya yang perlu dikonfigurasi adalah waktu tunda/*delay*. Waktu *delay* atau jarak adalah waktu yang digunakan untuk memberikan jarak antara file .bytes dan file .mp3/.wav, karena ketika file .bytes memberikan informasi *midi event*, adalah saatnya untuk proses instansiasi *note* dari atas bukan disaat *note* telah ditangkap. Maka jika file .mp3/.wav dan file .bytes bersamaan mulai diproses pada waktu yang sama, maka *note* dan file .mp3/.wav tidak akan bersifat sinkron dalam game ini. Maka waktu *delay* ini akan membuat .bytes dan mp3/wav bersifat sinkron dalam game ini, maskud dari sinkron adalah Midi File Handler tetap melakukan tugasnya dengan membaca file .bytes untuk membuat suatu *note*, sedangkan file .mp3/.wav akan berbunyi pada *timing* yang pas dengan *note* yang telah dibuat oleh Midi File Handler, sehingga ketika *note* tersebut datang, ketika *note* sudah sampai ke aktivator, suara dari file .mp3/.wav pun yang mewakili *note* tersebut juga berbunyi pada *timing* yang pas. Jarak antara *note* dengan aktivator dapat dilihat pada **Gambar 3.26**. Waktu jarak antar 2 files akan dalam bentuk angka *float*.



Gambar 3.26 *Ilustrasi* bagaimana *note* diinstansiasi hingga aktivator

- ***Channel***

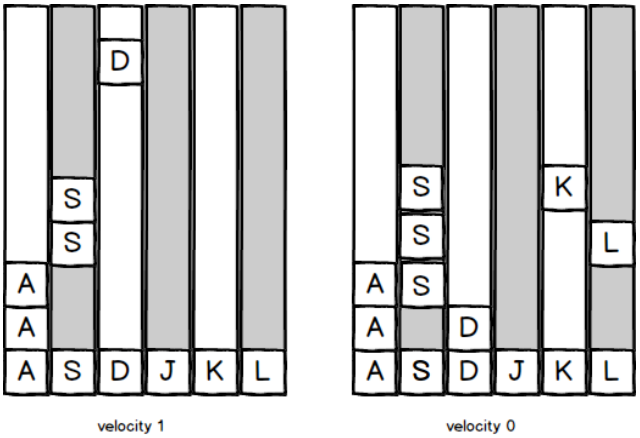
Selanjutnya konfigurasi dilanjutkan dengan memilih *channel* keberapa saja yang akan digunakan untuk membuat *note*. Karena tentu saja tidak semua *channel* akan dijadikan *Rhythm Game*, sebenarnya bagi Midi File Handler itu sendiri tidak akan berat dalam melakukan instansiasi *note* pada banyak *channel* namun apabila terlalu banyak *channel* maka akan membuat *note* yang akan diintansiasi, menjadi banyak namun sangat berantakan sehingga tidak layak untuk dimainkan. Maka perlu adanya pemilihan *channel* yang sesuai agar *note* yang dibuat teratur sesuai dengan nada-nada pada *channel* yang terpilih tersebut. Biasanya *channel* yang baik dalam *Rhythm Game* adalah *channel* yang memainkan melodi secara *ideal*, *note* yang terbentuk akan merepresentasikan *timing* melodi sehingga mudah diikuti oleh pemain. Namun *channel* yang memainkan ritme pun juga bisa membuat *note* yang baik, namun biasanya pada *Rhythm Game* pemain lebih mudah dalam mencerna melodi daripada ritme, selain itu nada/*noteOn* yang akan dihasilkan oleh *channel* ritme biasanya bersifat kompleks sehingga menghasilkan *note* yang susah dan banyak. Salah satu perbedaan *channel* dapat menyebabkan hal pada **Gambar 3.27**. Deklarasi *channel* dalam ranah programming tidak langsung menggunakan angka 1-16, untuk menentukan instrumennya ada suatu kode tersendiri agar *channel* tersebut dipanggil, *channel noteOn* dimulai dari 0x90 atau 144, inilah yang akan menjadi *channel noteOn* pada instrumen pertama. Jadi untuk menentukan *channel* instrumen maka akan menggunakan angka dari 144 – 159 untuk menentukan *channel* mana yang akan dimainkan.



Gambar 3.27 Perbedaan *Channel* yang dapat menyebabkan perbedaan jumlah *Note* yang terbentuk pada satu lagu.

- ***Velocity* Minimal**

Konfigurasi Juga dilakukan untuk menentukan *velocity* minimal, dimana *velocity* minimal berfungsi untuk menentukan nilai minimal dari *velocity*, biasanya ini digunakan untuk file midi jika file midi tersebut menggunakan *noteOn velocity* 0 sebagai *noteOff*(nada berhenti berbunyi). Oleh karena itu biasanya *velocity* minimal adalah 1 agar hanya pada *noteOn* yang memang bertujuan untuk *noteOn* saja yang akan memberikan perintah untuk instansiasi *note*(atau *noteOn* yang menghasilkan suatu bunyi). Namun terkadang pada beberapa midi file tertentu dapat menggunakan *noteOn velocity* 0 hanya untuk membuat *note* menjadi *double* (kondisi pada waktu tertentu keluar dua *note* sebaris sehingga perlu di tekan pada waktu yang sama) sehingga dapat sedikit mempersusah game, namun ini hanya bisa diaplikasikan dengan midi dengan *noteOn* durasi pendek (dimana *noteOn* dan *noteOff* bersamaan datang). Kondisi *double* yang disebabkan karena *velocity* minimal 0 dapat dilihat pada **Gambar 3.28**. Nilai pada *velocity* minimal adalah range angka antara 0 hingga 127 sesuai dengan ketentuan *midi event* dari midi general.



Gambar 3.28 Perbedaan *Velocity* minimal 0 dan 1 pada nada durasi pendek

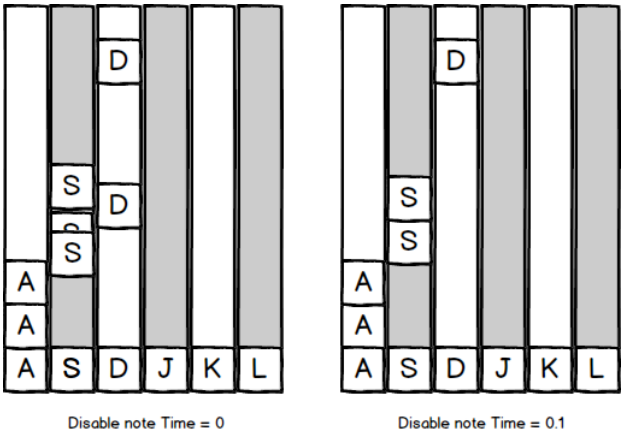
- **Tingkat Kesulitan**

Konfigurasi tingkat kesulitan tidak akan mempengaruhi proses instansiasi *note*. Namun ini berfungsi hanya untuk melengkapi *Interface*. Tingkat kesulitan hanya untuk memberi informasi kepada pemain tentang kesulitan pada suatu lagu.

- **Waktu *Limit Note***

Mengkonfigurasi waktu *limit note* berfungsi untuk menghentikan pekerjaan Midi File Handler sementara agar *note* tidak terbentuk pada saat berada di waktu *limit note*. Waktu *limit note* akan terjadi tiap kali *note* sudah dibentuk, tetapi waktu tersebut akan dipanggil hanya ketika waktu *limit note* lebih dari 0. Dengan waktu *limit note*, lagu midi dimana pada suatu *channel* memiliki *midi event* yang terlalu banyak atau terlalu dekat antara satu *note* dengan yang lain dapat direduksi, agar lagu pada saat permainan menjadi lebih mudah dan teratur. Hal ini dapat ditunjukan pada

Gambar 3.29 dengan asumsi lagu, *channel*, BPM serta waktunya sama.

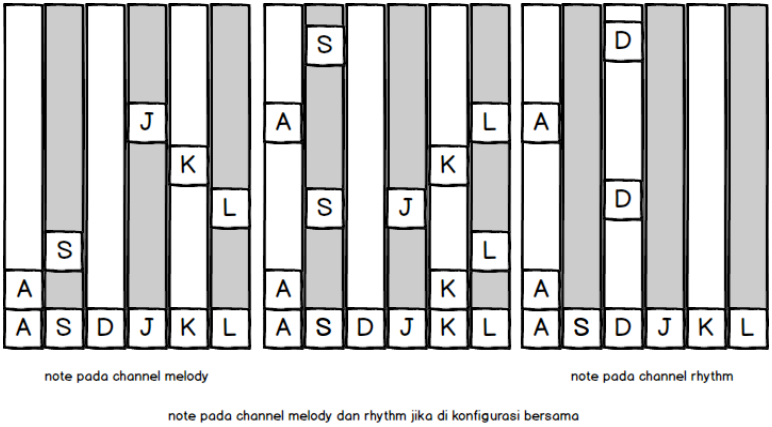


Gambar 3.29 Perbedaan Antara *Note Map* tanpa *Limit Note* dan dengan *Limit Note*

- ***Channel Melody* dan *Rhythm***

Konfigurasi ini hanya dilakukan untuk midi yang *channel*nya dikonfigurasi lebih dari satu, *channel* ini berfungsi untuk sebagai pembagi posisi *note*, ketika *channel* yang dikonfigurasi ternyata dimainkan dalam waktu yang sama, sehingga pemain masih dapat membedakan antara mana yang *melody* dan juga mana yang *rhythm*. Sehingga pemain tidak merasa bingung ketika *note* yang mewakili *channel melody* dan *note* yang mewakili *channel rhythm*, datang bersamaan. Apabila *channel melody* dan *channel rhythm* sudah di konfigurasi, maka setiap *note* yang mewakili *channel melody* akan dipindah ke 3 posisi bagian kanan yaitu J, K dan L sedangkan pada *channel rhythm* akan dipindah ke A, S dan D ketika kedua *note* tersebut datang pada waktu yang sama, terutama yang membentuk *note double*. Ilustrasi *note* dengan *channel* yang dikonfigurasi, dimainkan bersamaan adalah seperti pada **Gambar**

3.30. dalam aplikasi yang akan dirancang, konfigurasi ini akan bersifat opsional sehingga tidak perlu diisi apabila tidak diperlukan. Apabila konfigurasi ini tidak dibutuhkan maka cukup dengan mengisi “none” saja.



Gambar 3.30 *Note* pada *channel* melody dan rhythm (tengah) apabila di satukan.

- **Penyusunan File .ini untuk konfigurasi serta contohnya**

Setelah penjelasan apa saja yang perlu dikonfigurasi dalam kesatuan file midi (file .bytes dan file .mp3/.wav), pada subbab ini maka akan menjelaskan bagaimana cara membuat file .ini. Konfigurasi tiap lagu dapat dilakukan dengan membuat file .ini untuk mewakili konfigurasi dari satu kesatuan file midi. Semua file konfigurasi akan berisi dengan format sebagai berikut.

(Nama Lagu)|(Custom BPM)|(Waktu Delay/Tunda)|(*Channel*)|(*Velocity* Minimal)|(Tingkat Kesulitan)|(Waktu Disable *Note*)|(*Channel* melody)|(*Channel* Rhythm)

File .ini, file .bytes dan file .mp3/.wav nantinya harus memiliki nama yang sama agar dapat diakses oleh aplikasi game. Salah satu contoh dari file .ini akan dijelaskan juga pada subbab ini

beserta penjelasanya, contoh isi pada file .ini lainnya juga dapat dilihat pada **Lampiran 2**. Pada contoh di subbab ini akan menggunakan file .ini pada lagu yang berjudul Manuk Dadali, berikut isi dari file manukdadali.ini.

manukdadali|103.00|2.2|147|1|2|0.2|none|none

Dari isi file manukdadali.ini tersebut, dapat dijabarkan bahwa pada file midi Manuk Dadali akan dikonfigurasi sebagai berikut.

- "manukdadali" menunjukkan nama dari lagu, hal ini tidak mempengaruhi konfigurasi namun ini adalah awal dari file konfigurasi.
- "103.00" menunjukkan BPM / kecepatan dari lagu, nilai BPM ini diambil sesuai dengan nilai BPM dari lagu midi Manuk Dadali yang sebenarnya, agar *note* yang terinstansiasi akan sesuai dengan suara dari file .mp3/.wav yang akan dimainkan. Angka BPM ini akan ditampilkan pada halaman pemilihan lagu sebelum lagu dimulai.
- "2.2" menunjukkan waktu *delay* antara file ekstensi .bytes dengan file .mp3/.wav, waktu *delay* sebesar 2.2 detik tersebut akan membiarkan Midi File Handler untuk memulai *track sequencing* pada *midi event* terlebih dahulu agar *note* dibuat sebelum akhirnya setelah 2.2 detik tersebut berlalu, file .mp3/.wav akan dibunyikan. Sehingga *note* akan sampai menuju aktivator sesuai dengan suara musik yang tepat.
- "147" menunjukkan *channel* yang akan digunakan dalam membuat *note*, 147 dapat berarti bahwa *channel* yang akan dimainkan adalah *channel* instrumen ke 4 (144 menunjukkan *channel* ke 1, 145 menunjukkan *channel* ke 2 dan sebagainya). Maka *note* yang akan diinstansiasi adalah berdasarkan *midi event* pada *channel* instrumen ke 4.
- "1" menunjukkan ketentuan berapa *velocity* minimal pada *midi event* sehingga *note* dapat diinstansiasi, ketentuan tersebut bernilai 1 sehingga apabila suatu *midi event* memiliki *velocity* kurang dari 1 maka *note* tidak akan diinstansiasi bahkan

walaupun *midi event* tersebut sudah memiliki *channel* 147 atau *channel* yang sudah sesuai dengan konfigurasi.

- "2" hanya menunjukkan tingkat kesulitan pada lagu tersebut, hal ini tidak akan mempengaruhi bagaimana *note* dengan lagu pada permainan nantinya, angka ini hanya menunjukkan level pada lagu tersebut. Angka ini akan ditampilkan bersamaan dengan BPM pada saat pemilihan lagu sebagai tingkat kesulitan.
- "0.2" menunjukkan waktu *delay note*, nilai waktu ini berfungsi untuk menghentikan kinerja instansiasi *note* bahkan walaupun *midi event* sudah sesuai ketentuan. Waktu delay *note* ini akan dimulai setiap satu *note* sudah berhasil dibuat. Setelah waktu *delay note* berlangsung selama 0.2 detik, tidak akan ada instansiasi *note* dan ketika *delay note* sudah berakhir maka *note* dapat dibuat kembali. Hal ini mengakibatkan pengurangan terhadap jumlah *note* karena tidak adanya *note* yang berjarak lebih dekat daripada *note* yang dibuat setelah 0.2 detik *note* pertama diinstansiasi, sehingga jumlah *note* berkurang dan permainan menjadi lebih mudah.
- "none" yang berjumlah dua pada isi file manukdadali.ini tersebut, menunjukkan bahwa tidak ada *channel* yang akan dipisah sebagai *channel* ritme dan *channel* melodi, sehingga posisi *note* pada *channel* apapun akan selalu pada range 1 hingga 6.

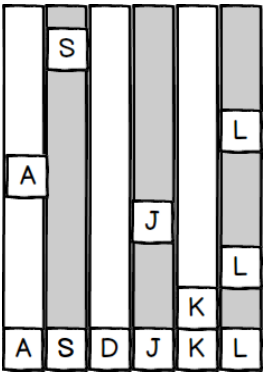
3.2.6.2 Perancangan variasi *note*

Setelah melakukan perancangan data midi dan konfigurasinya sehingga *note* akan sesuai dengan lagu setidaknya, maka hal selanjutnya yang dapat dirancang adalah variasi *note*, variasi *note* berfungsi agar membuat permainan menjadi lebih bisa dinikmati dan mempermudah pemain untuk mengerti melodi pada lagu. Variasi *note* akan dipengaruhi oleh angka *note* yang diperoleh dari *midi event*, kecuali untuk membentuk *double note* dimana *double note* hanya sebagai antisipasi terjadinya tabrakan *note* agar *note* yang ada lebih dari satu tidak menumpuk pada posisi kolom yang sama, karena hal ini menyebabkan tabrakan kedua *note*

tersebut, akibatnya pemain tidak memahami bahwa adanya lebih dari satu *note* pada satu kolom yang sama, sehingga satu *note* akan luput walaupun pemain berhasil menangkap.

- ***Arbitrary Note***

Susunan *note* ini akan terjadi kondisi *default*, tidak memenuhi kondisi variasi *note* lainnya. Arbitrary *note* tidak memiliki hubungan antar *note*, sehingga *note* selanjutnya akan terbentuk pada posisi kolom mana saja secara random. *Arbitrary note* dapat diilustrasikan seperti **Gambar 3.31**.



Gambar 3.31 Susunan Arbitrary *note* yang mengacak *note* ke sembarang posisi

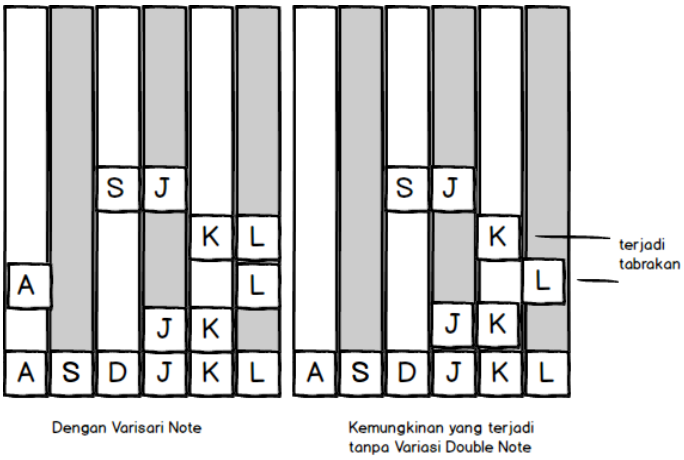
- ***Double Note***

Double note merupakan bentuk antisipasi supaya tidak terjadi tabrakan pada satu kolom yang sama. *Double note* terjadi karena kondisi disaat *note* diinstansiasi pada waktu yang benar-benar sama, sehingga *note* akan datang sebaris. Apabila terjadi instansiasi *note* pada waktu yang sama tersebut, terkadang bisa terjadi *note* tersebut ada di posisi kolom yang sama sehingga seakan-akan *note* tetap satu dalam satu baris, namun ternyata ada 2 dalam posisi

kolom yang sama sehingga tidak terlihat. *Double note* akan mengantisipasi tabrakan *note* tersebut dengan cara menggeser *note* tersebut ke posisi lainya. Berikut perkondisian double *note*.

- Apabila *note* yang pertama berada di A maka *note* yang disampingnya akan dirandom antara S, D, J, K, L.
- Apabila *note* yang pertama berada di S maka *note* yang disampingnya akan dirandom antara A dan J.
- Apabila *note* yang pertama berada di D maka *note* yang disampingnya akan dirandom antara A, S, J dan K.
- Apabila *note* yang pertama berada di J maka *note* yang disampingnya akan dirandom antara S, D, K dan L.
- Apabila *note* yang pertama berada di K maka *note* yang disampingnya akan dirandom antara J dan L.
- Apabila *note* yang pertama berada di L maka *note* yang disampingnya akan dirandom antara A, S, D, J dan K .

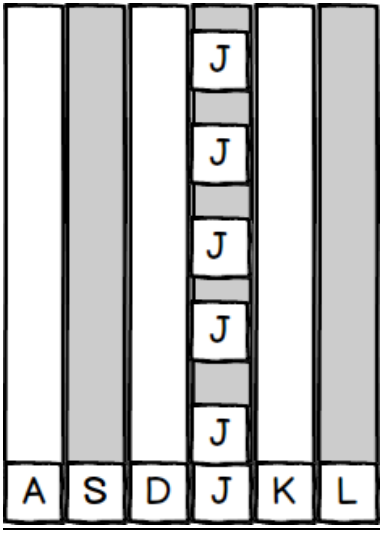
Dengan perkondisian tersebut *note* tidak akan pada kolom yang sama apabila *note* tersebut sebaris. Ilustrasi *double note* dapat dilihat pada **Gambar 3.32**.



Gambar 3.32 Perbandingan antara dengan *Double Note* dan dengan yang tidak

- ***Repeat Note***

Repeat note adalah kondisi yang akan terjadi ketika *note* selanjutnya adalah *note* yang sama. sehingga menghasilkan susunan *note* dengan kolom yang sama. Dengan ini *Repeat note* akan memberikan kesan seakan-akan mengulang *note* yang sama. Ilustrasi *Repeat note* dapat dilihat pada **Gambar 3.33** dengan asumsi *note* tersebut adalah 60, 60, 60 , 60, 60.



Gambar 3.33 Ilustrasi *Repeat Note*

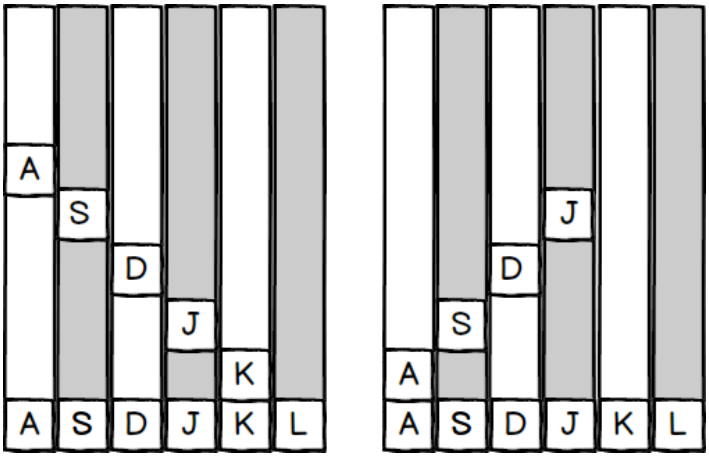
- ***Ladder Note***

Ladder note adalah variasi *note* yang seakan-akan membentuk sebuah tangga dari kiri kekanan, menunjukkan bahwa *note* selanjutnya semakin tinggi, namun *ladder note* akan dibatasi selisihnya yaitu 20 apabila lebih dari itu maka *note* selanjutnya akan tetap menjadi *arbitrary note* karena sudah memiliki kesan melompat nada.

- ***Reversed Ladder Note***

Reversed ladder note adalah Variasi *note* yang merupakan kebalikan dari *ladder note*, menunjukkan bahwa *note* selanjutnya semakin rendah. *Reversed ladder note* juga akan di batasi selisihnya yaitu 20 dan jika lebih dari itu maka *note* selanjutnya akan menjadi *arbitrary note* Juga Contoh Ilustrasi *Ladder note* dengan *reversed ladder note* ditunjukkan pada **Gambar 3.34**.

Urutan nada pada *reversed ladder note*: 100, 85, 75, 60, 53.
Urutan nada pada *ladder Note*: 30, 33, 40, 59.



Gambar 3.34 Ilustrasi Reversed Ladder *Note* dan Ladder *Note*

3.2.7 Perancangan Pembuat *Note* dengan Midi File Handler

Dari awal tujuan dari aplikasi ini adalah membuat sebuah *Rhythm Game* yang bersifat otomatis, dimana yang dimaksud dengan otomatis adalah membuat susunan *note map* bersamaan dengan saat lagu dimainkan. Berdasarkan use case UC-006, terdapat sebuah langkah dimana sistem akan menginstansiasi *note*, maka pada subbab ini hal ini akan di bahas.

Dalam pembuatan *note* melalui *track sequencer* pada Midi File Handler akan melibatkan hal yang sudah dibahas pada **Subbab 3.3.6**, dimana hal-hal tersebut akan diterapkan dan digunakan dalam pembuatan *note*. Dimana pada file .ini, Midi File Handler akan membutuhkan ketetentuan *channel* apa saja yang akan dibuat *note*nya serta minimal *velocity* pada setiap *midi event* yang datang berapa agar *note* dapat dibuat. Diluar file .ini, nada pada setiap *midi event* akan memutuskan posisi *note* selanjutnya.

Jadi dalam memutuskan apakah *note* akan dibuat atau tidak setiap *midi event* datang adalah melalui tahap berikut.

1. Mengambil satu set *midi event message* berupa *channel*, *note* (nada) dan *velocity*.
2. Cek apabila *channel* termasuk *channel* yang sudah ditentukan pada file konfigurasi dan *velocity* lebih besar sama dengan yang ditentukan pada file konfigurasi, apabila iya maka lanjut ke tahap berikutnya.
3. Cek apabila masih berada pada masa *limit note* (apabila pada file konfigurasi terdapat waktu *limit note*).
4. Membuat variabel randomizer dengan nilai random antara 1-6.
5. Apabila *note* (nada) dan *velocity* sama dengan variabel temp_note dan temp_velocity (pada midievent sebelumnya) maka akan melakukan *repeat note* dengan mengubah randomizer sama dengan temp.
6. Apabila *double note* (variabel randomizer sama dengan temp variabel posisi *note* sebelumnya pada waktu yang bersamaan), maka geser randomizer antara 1-5 selisih (tergantung posisi dari I) dari variabel temp (bernilai posisi dari *note* sebelumnya).
7. Apabila *note*(nada) dari *midi event* memiliki selisih maksimal 20 lebih besar dari temp_note (variabel *note* sebelumnya) maka akan mengubah randomizer menjadi temp + 1 apabila randomizer tidak bernilai 6 (posisi pojok kanan), apabila randomizer bernilai 6 maka randomizer akan diubah menjadi 1.
8. Apabila *note*(nada) dari *midi event* memiliki selisih maksimal 20 lebih kecil dari temp_*note*(variabel *note* sebelumnya) maka akan

mengubah randomizer menjadi temp – 1 apabila randomizer tidak bernilai 1 (posisi pojok kiri), apabila randomizer bernilai 1 maka randomizer akan diubah menjadi 6.

9. Simpan nilai randomizer menjadi nilai temp baru (temp = randomizer).
10. Simpan nilai *note* menjadi nilai temp_note baru(temp_note = *note*).
11. Simpan nilai *velocity* menjadi nilai temp_velocity baru(temp_velocity = *velocity*).
12. Apabila randomizer bernilai 1 maka instansiasi *note* A.
    Apabila randomizer bernilai 2 maka instansiasi *note* S.
    Apabila randomizer bernilai 3 maka instansiasi *note* D.
    Apabila randomizer bernilai 4 maka instansiasi *note* J.
    Apabila randomizer bernilai 5 maka instansiasi *note* K.
    Apabila randomizer bernilai 6 maka instansiasi *note* L.
13. Selesai.

Tahap ini akan diilustrasikan dalam bentuk pseudocode sebagai berikut.

```
FOREACH m ON midieventmessage
 IF set channel contains m.channel and m.velocity bigger than
 min_velocity
  IF delaynote IS true
   RETURN
  ENDIF
  DECLARE randomizer AS INTEGER
  SET randomizer TO CALL random_number_generator_function(1, 6)
  //double note
  IF randomizer EQUAL TO temp
   DECLARE mover AS INTEGER
   DECLARE operator AS INTEGER
   IF randomizer EQUAL TO 6
    SET mover TO CALL random_number_generator_function(1,5)
    SET operator TO 1
   ELSE IF randomizer EQUAL TO 5 OR 2
    SET mover TO 1
    SET operator TO CALL random_number_generator_function(1,2)
   ELSE IF randomizer EQUAL TO 3 OR 4
    SET mover TO CALL random_number_generator_function(1,2)
    SET operator TO CALL random_number_generator_function(1,2)
   ELSE IF randomizer EQUAL TO 1
```

```
    SET mover TO CALL random_number_generator_function(1,5)
    SET operator TO 1
   ENDIF
  //repeat note
  IF m.note EQUAL TO temp_note and m.velocity equal to
  temp_velocity
   SET randomizer TO temp
  ENDIF
   IF operator EQUAL TO 1
    Substract randomizer with mover
   ELSE
    ADD randomizer with mover
   ENDIF
  ENDIF
  //ladder note
  IF m.note BETWEEN temp_note+1 and temp_note + 20
   IF m.note IS SMALLER than 6
     ADD randomizer with 1
   ELSE
     SET randomizer to 1
   ENDIF
  ENDIF
  //reversed ladder note
  IF m.note BETWEEN temp_note-20 and temp_note
   IF m.note IS BIGGER than 1
     SUBSTRACT randomizer with 1
   ELSE
     SET randomizer to 6
   ENDIF
  ENDIF
 //Finally Instantiate the note
  IF randomizer EQUALS TO 1
   DO instantiate noteA
  ELSE IF randomizer EQUALS TO 2
   DO instantiate notes
  ELSE IF randomizer EQUALS TO 3
   DO instantiate noteD
  ELSE IF randomizer EQUALS TO 4
   DO instantiate noteJ
  ELSE IF randomizer EQUALS TO 5
   DO instantiate noteK
  ELSE IF randomizer EQUALS TO 6
   DO instantiate noteL
  ENDIF
 ENDIF
ENDFOREACH
```

(Halaman Ini Sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan perangkat lunak. Di dalamnya mencakup proses penerapan dan pengimplementasian dalam bahasa C# untuk implementasi kode dan antar muka yang mengacu pada rancangan yang telah dibahas sebelumnya akan menggunakan Unity Editor.

4.1 Lingkungan Implementasi

Lingkungan implementasi dari tugas akhir dijelaskan pada **Tabel 4.1**.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

Perangkat Keras		Perangkat Lunak	
Prosesor	Memori	Sistem Operasi	Perangkat Pengembang
Intel Core-i5 4210	4 GB RAM	Windows 10 Profesional 64 bit	Unity Engine 2017.2.2f1 Visual Studio 2017

4.2 Implementasi Permainan

Implementasi terdapat dua macam yakni implementasi kasus penggunaan dan antarmuka, untuk antarmuka langsung dapat dilakukan *screenshooting,* sedangkan untuk kasus penggunaan hanya diambil beberapa *source code* yang dianggap mewakili dari keseluruhan yang mana terdapat pada, **Lampiran 1 .**

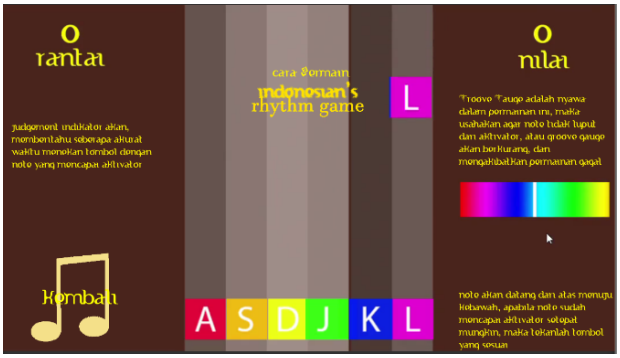
4.2.1 Implementasi Tampilan Main Menu

Nama *scene* yang mewakili tampilan ini adalah mainMenu.unity, tampilan ditunjukkan pada **Gambar 4.1**.

Gambar 4.1 Tampilan *Main Menu*

4.2.2 Implementasi Tampilan *How To Play* (Hanya menampilkan mp4 saja)

Nama *scene* yang mewakili tampilan ini adalah howtoplay.unity, tampilan ditunjukkan pada **Gambar 4.2**.

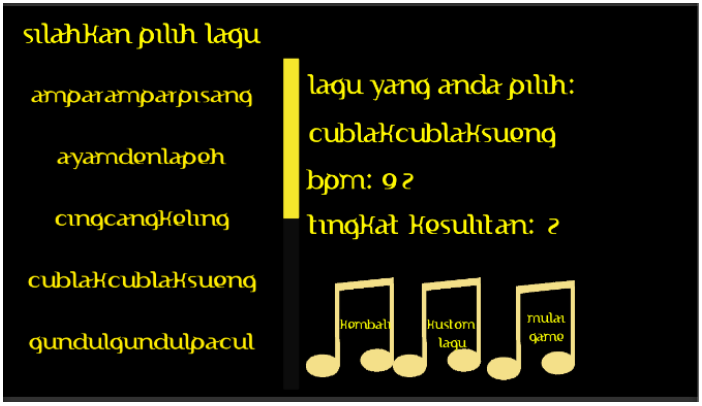


Gambar 4.2 Tampilan *How To Play*

4.2.3 Implementasi Tampilan Pemilihan Lagu

Nama *scene* yang mewakili tampilan ini adalah musicselection.unity. Tampilan ini ditunjukkan pada **Gambar 4.3** lalu **Gambar 4.4,** setelah menekan salah satu lagu pada daftar lagu.

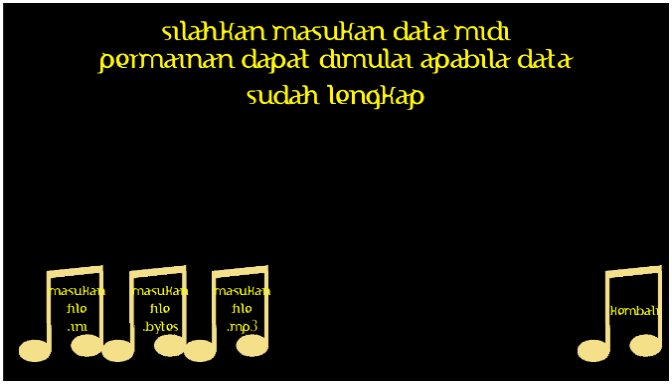
Gambar 4.3 Tampilan Pemilihan Lagu sebelum memilih salah satu lagu



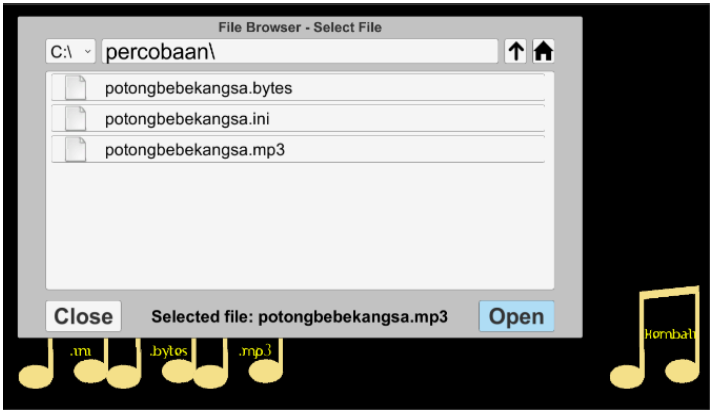
Gambar 4.4 Tampilan Pemilihan Lagu setelah memilih salah satu lagu

4.2.4 Implementasi Tampilan Kustomisasi Musik

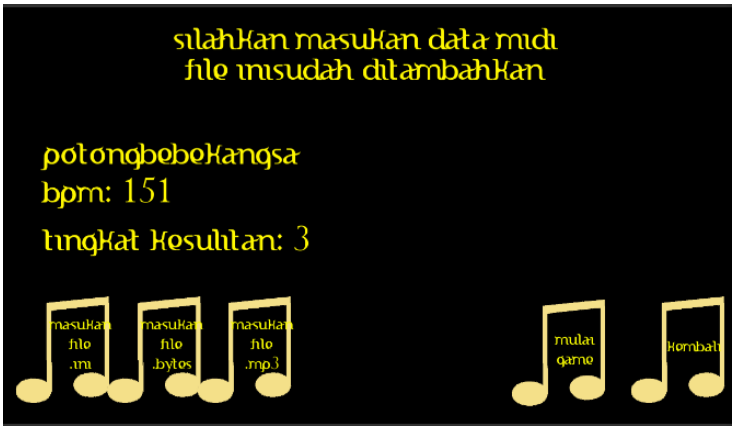
Nama *scene* yang mewakili tampilan ini adalah musicCustomization.unity. Tampilan ini akan ditunjukkan pada **Gambar 4.5**, **Gambar 4.6** dan **Gambar 4.7**.



Gambar 4.5 Tampilan Lagu Kustomisasi sebelum menambah file .bytes, file .wav dan file .ini



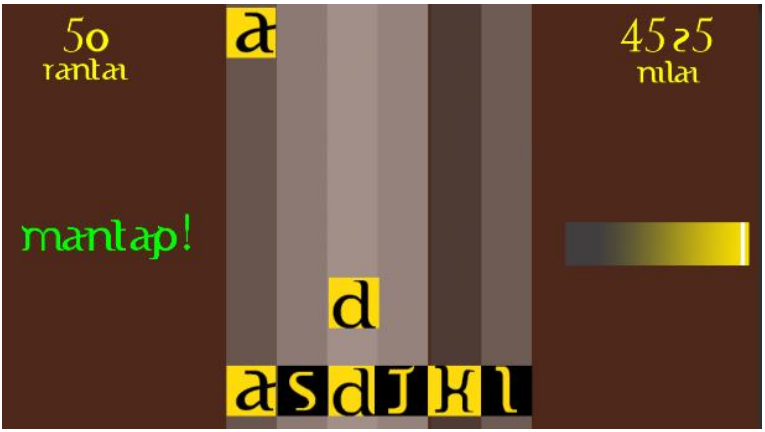
Gambar 4.6 Tampilan Lagu Kustomisasi saat open file untuk file .bytes, file .wav dan file .ini



Gambar 4.7 Tampilan Kustomisasi Musik setelah data lengkap

4.2.5 Implementasi Tampilan Main Game

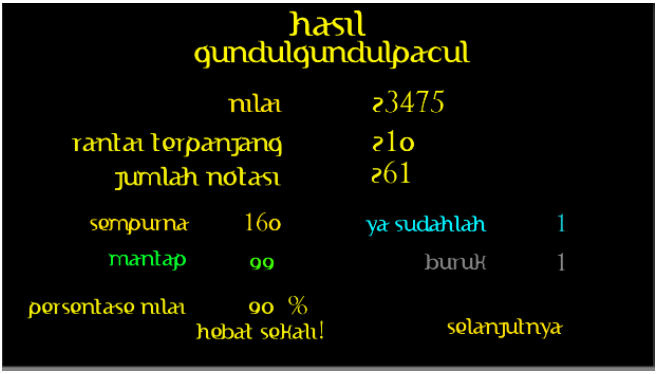
Nama *scene* yang mewakili tampilan ini adalah theGame.Unity. Tampilan ini akan ditunjukkan pada **Gambar 4.8**.



Gambar 4.8 Tampilan Main Game

4.2.6 Implementasi Tampilan Hasil

Nama *scene* yang mewakili tampilan ini adalah results.Unity. Tampilan ini akan ditunjukkan pada **Gambar 4.9**.



Gambar 4.9 Tampilan Hasil

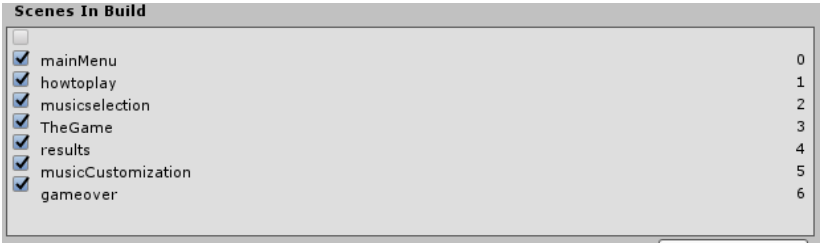
4.2.7 Implementasi Tampilan Gagal/"sayang sekali"

Nama *scene* yang mewakili tampilan ini adalah gameover.Unity. Tampilan ini akan ditunjukkan pada **Gambar 4.10**.

Gambar 4.10 Tampilan gagal/"sayang sekali"

4.2.8 Implementasi Kasus Penggunaan

Dari delapan kasus penggunaan implementasi disajikan dalam *source code* dengan nama file .cs yang sebenarnya, dimana versi lengkap masing-masing file .cs tersebut untuk kasus penggunaan terdapat pada **Lampiran 1.** dalam Unity, kode disajikan terpisah tiap komponenya jadi kemungkinan pada satu use case bisa mewakili lebih dari satu kode. Pada ButtonFunction.cs yang akan dilakukan adalah hanya memnindahkan antar *scene* saja dimana Built *Scene* Listnya terdapat pada **Gambar 4.11**.



Gambar 4.11 Daftar *Scene*s in build serta angka *Scene*nya

1. **Implementasi Kasus Penggunaan Masuk ke Pemilihan Lagu**

   - Pemain menekan tombol “Mulai Bermain”.
     tombol tersebut memanggil fungsi berikut.
     `ButtonFunction.Load`*`Scene`*`(2);`
   - Sistem menampilkan Daftar Lagu.
     Setelah mendapatkan panggilan fungsi maka, sistem langsung mengakses fungsi Load*Scene*() script ButtonFunction.cs, dimana a = 2.

```
public void LoadScene(int a)
 {
     if(a == 2|| a== 5)
     {
         emptythesongs();
     }
     SceneManager.LoadScene(a);
 }
```

```
    //mengosongkan terpilih
    void emptythesongs()
    {
        PlayerPrefs.SetString("songName", "");
        PlayerPrefs.SetString("songBPM", "");
        PlayerPrefs.SetString("songLevel", "");
        PlayerPrefs.SetString("Custom.ini", "");
        PlayerPrefs.SetString("Custom.bytes", "");
        PlayerPrefs.SetString("Custom.wav", "");
    }
```

Setelah fungsi tersebut, maka sistem sudah menampilkan halaman pemilihan lagu, lalu sistem akan mengambil list lagu dengan fungsi start (otomatis dijalankan) pada initiator.cs.

```
void Start () {
GameObject Button = Resources.Load("SongLabel") as
GameObject;
string filePath = Application.dataPath +
"/Resources/Midi Notes";
DirectoryInfo f = new DirectoryInfo(filePath);
foreach (var line in f.GetFiles("*.bytes"))
{
    GameObject Par;
    Par = GameObject.Find("GridWidthElement");

    GameObject butt = Instantiate(Button);
    butt.name = line.Name.Replace(".bytes", "");
    butt.transform.SetParent(Par.transform, false);
     butt.GetComponentInChildren<Text>().text =
     line.Name.Replace(".bytes", "");
}
}
```

2. Implementasi Kasus Penggunaan Masuk Ke Halaman Cara Bermain

- Pemain menekan tombol “Cara Bermain”.
  tombol tersebut memanggil fungsi berikut.
  `ButtonFunction.Load*Scene*(1);`
- Sistem menampilkan halaman Cara Bermain.
  Setelah mendapatkan panggilan fungsi maka, sistem langsung mengakses fungsi Load*Scene*() dalam *source code* ButtonFunction.cs, dimana a = 1.
  `public void Load*Scene*(int a)`

```
{
    if(a == 2|| a== 5)
    {
        emptythesongs();
    }
    SceneManager.LoadScene(a);
}
```

- Sistem memulai video tutorial, langkah ini dilakukan secara automatis dimana video tutorial sudah bersifat *play on awake*.

3. Implementasi Kasus Penggunaan Memilih Lagu

- Pemain mengeklik salah satu lagu yang diinginkan.
  tombol tersebut akan memanggil fungsi berikut. `musicinfosetter.setMusicInfo();`
- Sistem menampilkan Informasi BPM dan Tingkat Kesulitan.
- Sistem menampilkan Tombol Mulai.
  Setelah mendapatkan panggilan fungsi maka, sistem langsung mengakses fungsi setMusicInfo() pada musicinfosetter.cs.

```
public void setMusicInfo(){
            name = EventSystem.current.currentSelectedGameObject.name;
            PlayerPrefs.SetString("songName", name);
            string filePath = Application.dataPath + "/Resources/MidiConfig/"
+ name + ".ini";
            string config = File.ReadAllLines(filePath).First()
            string[] entries = config.Split('|');
            if (entries[0] == name)
            {
                string[] bpmFixed = entries[1].Split('.');
                PlayerPrefs.SetString("songBPM", "bpm: " + bpmFixed[0]);
                PlayerPrefs.SetString("songLevel","tingkat kesulitan: " +
entries[5]);
             }
              //setelah itu instantiate tombol untuk start
              if(!instantiated)
              {
                  GameObject ParentCanvas = GameObject.Find("Canvas");
                  GameObject button;
                  button = Instantiate(Resources.Load("TombolMulai")) as
            GameObject;
```

```
                button.transform.SetParent(ParentCanvas.transform, false);
                instantiated = true;

            }
        }
```

4. Implementasi Kasus Penggunan Membuka File Lagu dari sumber eksternal

- Pemain menekan tombol ‘Kustomisasi lagu’.
  tombol tersebut akan memanggil fungsi berikut.
  `ButtonFunction.Load*Scene*(5);`
- Sistem menampilkan Halaman kustomisasi Lagu.
  Setelah mendapatkan panggilan fungsi maka, sistem langsung mengakses fungsi Load*Scene*() dalam *source code* ButtonFunction.cs, dimana a = 5.

```
public void Load Scene(int a)
{
    if(a == 2|| a== 5)
    {
        emptythesongs();
    }
    Scene Manager.Load Scene(a);
}
```

- Pemain klik membuka file .wav.
  Tombol tersebut akan memanggil fungsi berikut.
  `FileBrowserView.OpenFileBrowser();`
- Sistem menampilkan File Browser untuk mencari file .wav, file .bytes dan file .ini.
  Dimana gameObject adalah FileBrowser Masing-masing typefile.

```
public void OpenFileBrowser(){
     gameObject.SetActive(true);
}
```

- Pemain klik open pada file browser ini dilakukan setelah Pemain klik salah satu file dengan tipe yang sesuai. Lalu memanggil fungsi berikut.
  `FileBrowserView.OpenFileButtonClick();`
- Sistem mengambil file .bytes, file .wav atau file .ini eksternal tersebut dari folder eksternal pada fungsi

openFileButtonClick() yang sedikit dimodifikasi dari FileBrowserView.cs (referensi: https://github.com/poisins/Unity3DFileBrowser).
Filetype akan berisi *file extension* yang sudah ditentukan masing masing file browser.

```
public void OpenFileButtonClick(){
     if(!string.IsNullOrEmpty(OpenFilePath) &&
     FileBrowser.BrowserCallback != null)
             FileBrowser.BrowserCallback
             (OpenFilePath);
     if(OpenFilePath.Contains(filetype)){
             PlayerPrefs.SetString("Custom." +
             filetype, OpenFilePath);
             PlayerPrefs.SetString("stats", "file " +
             filetype + "sudah ditambahkan");
     }
     else{
         PlayerPrefs.SetString("stats", "mohon untuk
        memasukan data yang benar");}
        }
```

Lalu secara otomatis sistem akan menjalankan fungsi update pada script musicinfosetterforcustoms.cs (wavfile, bytesfile dan inifile akan berisi path file yang sudah di open oleh pemain).

```
inifile =PlayerPrefs.GetString("Custom.ini");
bytesfile = PlayerPrefs.GetString("Custom.bytes");
wavfile = PlayerPrefs.GetString("Custom.wav");
```

- Pemain klik close.
  Ketika pemain klik close pada file browser, masing - masing tombol close pada file browser yang berbeda akan memanggil fungsi yang berbeda. tiga tombol close pada file browser yang berbeda akan memanggil fungsi berikut.

```
FileBrowserView.CloseFileBrowser();
```

  Lalu khusus FileBrowserWindowWav, akan memanggil juga fungsi berikut.

```
musicinfosetterforcustoms.setName();
```

  Sedangkan khusus FileBrowserWindowIni, akan memanggil juga fungsi berikut.

```
musicinfosetterforcustoms.setSongInfo();
```

- Sistem menampilkan nama lagu.

Diwakilkan oleh fungsi setName dari musicinfosetterforcustoms.cs.

```
public void SetName()
{
  namewithtype = wavfile.Split('\\').Last();
  name = namewithtype.Replace(".wav","");
  PlayerPrefs.SetString("songName", name);
}
```

- Sistem menampilkan BPM serta tingkat kesulitan pada lagu.
  Diwakilkan oleh fungsi setSongInfo dari musicinfosetterforcustoms.cs.

```
public void setSongInfo()
{
     filepath = PlayerPrefs.GetString("Custom.ini");
     string config =
     File.ReadAllLines(filepath).First();
     string[] entries = config.Split('|');
     string[] bpmFixed = entries[1].Split('.');
     PlayerPrefs.SetString("songBPM", "bpm: " +
     bpmFixed[0]);
     PlayerPrefs.SetString("songLevel", "tingkat
     kesulitan: " + entries[5]);
}
```

- Sistem menampilkan tombol “Mulai”.
  Aktivitas ini memanggil juga fungsi update dari musicinfosetter for customs, ketika semua file telah diupload.

```
if (bytesfile != "" && inifile != "" && wavfile != ""
&& !instantiated) {

     GameObject ParentCanvas =
     GameObject.Find("Canvas");
     GameObject button;
     button = Instantiate (Resources.Load
     ("TombolMulai")) as GameObject;
     button.transform.SetParent(ParentCanvas.transfor
     m, false);
     instantiated = true;
}
```

5. Implementasi Kasus Penggunaan Kembali Ke Menu

- Pemain menekan/memilih tombol Kembali. `ButtonFunction.Load*Scene*(0)`
- Sistem kembali ke Menu Utama.
  Pada ButtonFunction.cs Dimana a = 0.

```
public void Load Scene(int a)
{
    if(a == 2|| a== 5)
    {
        emptythesongs();
    }
    Scene Manager.Load Scene(a);
}
```

6. Implementasi Kasus Penggunaan Memainkan Lagu dan Menekan *Note*

- Pemain menekan/memilih tombol "Mulai". `ButtonFunction.Load*Scene*(3);`
- Sistem menampilkan halaman permainan yang segera dimulai.
  Pada ButtonFunction.cs Dimana a = 3.

```
public void Load Scene(int a)
{
    if(a == 2|| a== 5)
    {
        emptythesongs();
    }
    Scene Manager.Load Scene(a);
}
```

- Sistem akan memunculkan *note* selama lagu berlangsung kasus ini akan masuk pada *source code* TestSequencer.cs dimana *source code* tersebut menjadi tempat dimana *Track Sequencer* melakukan *sequencing* terhadap *midi event* yang telah disusun oleh Midi File Handler, TestSequencer akan menggunakan class smflite untuk mengeluarkan midievent dari file bytes yang dikonversi dari file midi.

- Sebelum *note* mulai di instansiasi, yaitu menyiapkan semua pada file untuk proses running nanti, melalui fungsi start() pada TestSequncer.cs.

```
IEnumerator Start (){
//menentukan sumber prefab dari note
 noteA = Resources.Load("note-A") as GameObject;
 notes = Resources.Load("note-S") as GameObject;
 noteD = Resources.Load("note-D") as GameObject;
 noteJ = Resources.Load("note-J") as GameObject;
 noteK = Resources.Load("note-K") as GameObject;
 noteL = Resources.Load("note-L") as GameObject;
 played = false;
 temp = 0;
 string Name = PlayerPrefs.GetString("songName");
 if (PlayerPrefs.GetString("Custom.ini") != ""){
 filepath = PlayerPrefs.GetString("Custom.ini");
     }
  else filepath = Application.dataPath +
 "/Resources/MidiConfig/" + Name + ".ini";
  if (PlayerPrefs.GetString("Custom.wav") != "")
  {
     string CustomAudio = PlayerPrefs.GetString(
"Custom.wav");
     UnityWebRequest www =
UnityWebRequestMultimedia.GetAudioClip("file://" + CustomAudio,
AudioType.WAV);
     yield return www.Send();

     AudioClip clipper =
     DownloadHandlerAudioClip.GetContent(www);
     GetComponent<AudioSource>().clip = clipper;
  }
  else
     GetComponent<AudioSource>().clip =
Resources.Load("MidiAudio/" + Name) as AudioClip;
     if (PlayerPrefs.GetString("Custom.bytes")
     != "")
     {
        sourceAlternative = PlayerPrefs.
        GetString("Custom.bytes");
         custom = File.
        ReadAllBytes(sourceAlternative);
     }
     else {
         sourceFile = Resources.Load
("Midi Notes/" + Name) as TextAsset;
     }
```

```
            string config = File.ReadAllLines
(filepath).First();
            string[] entries = config.Split('|');
           //inisialisasi data
              if (entries[0] == Name){
                              bpm =
      float.Parse(entries[1]);
               //waktu tunda/delay
               timedelay = float.Parse(entries[2]);
               //channel
               string[] channels = entries[3].
               Split(',');
               int i = 0;
               //memasukan channel ke dalam list
               curstat = new List<byte>(new byte[] {
               });
               curmelodystat = new List<byte>(new
               byte[] { });
               currhythmstat = new List<byte>(new
               byte[] { });
               foreach (var channel in channels)
               {
                curstat.Add(byte.Parse(channels[i]));
                i++;
               }
               //velocity minimal
               curvelomin = byte.Parse(entries[4]);
               //waktu limit note
               notelimittime = float.Parse(entries[6]);
               //jika ada dua channel yang dimainkan
              bersamaan
               //channel melodi dan ritme
              if (entries[7] !="none" && entries[8] !=
              "none")
               {
                 mainmelody = byte.Parse(entries[7]);
                 rhythm = byte.Parse(entries[8]);
               }
      }
       //untuk mengambil bytes file menuju
       midifileloader (class pada library smflite)
      song = MidiFileLoader.Load(sourceFile.bytes)
      yield return new WaitForSeconds (1.0f);
       //fungsi untuk memulai lagu karena sudah siap
       semua
      ResetAndPlay ();
      notecount = 0;}
```

- Menginstansiasi *note* selama lagu berlangsung.

Dengan mengakses fungsi applymessages pada testsequencer.cs, disini juga variasi *note* pada **Subbab 3.3.6.2** diterapkan.

```
foreach (var m in messages) {
 if ( curstat.Contains(m.status) &&
 m.data2>=curvelomin)
 {
 //Apabila masih pada limit
   if(TemporaryLimit)
   {
    return;
   }
   //randomizer menentukan posisi kolom
   //Arbitrary note diawal
   int randomizer= Random.Range(1, 7);
   if (mainmelody != null && rhythm != null){
        if (m.status == mainmelody){
        randomizer = Random.Range(4, 7);}
        else if (m.status == rhythm) {
        randomizer = Random.Range(1, 4);
        }}
   //memulai proses Waktu Limit note
if (notelimittime > 0) {
StartCoroutine(limitNote());}
//Double Note agar tidak ada note yang tabrakan
jika ada note double/ note yang datang bersamaan
if (randomizer == temp){
 //pada channel melody
 if(m.status == mainmelody) {
   if (randomizer == 6){
     int mover = Random.Range(1, 2);
     randomizer -= mover;
   }
   else if (randomizer == 5 ) {
     int mover = 1;
     int operators = Random.Range(1, 2);
     if (operators == 1)
     { randomizer -= mover; }
     else{randomizer += mover; }
   }
    else if (randomizer == 4){
      int mover = Random.Range(1, 2);
        randomizer += mover; }
 }
 //pada channel rhythm
 else if (m.status == rhythm)
{
 if (randomizer == 3)
```

```
 {
   int mover = Random.Range(1, 2);
   randomizer -= mover; }
 else if (randomizer == 2) {
   int mover = 1;
   int operators = Random.Range(1, 2);
   if (operators == 1) {
     randomizer -= mover;}
   else{
     randomizer += mover; }
 }
 else if (randomizer == 1) {
   int mover = Random.Range(1, 2);
   randomizer += mover; }
}
//tanpa channel melody dan rhythm
else{
 if (randomizer == 6) {
    int mover = Random.Range(1, 5);
    randomizer -= mover;
    }
 else if (randomizer <= 5 && randomizer >= 2){
    int mover = 0;
    if (randomizer == 5 || randomizer == 2){
        mover = 1;}
    if (randomizer == 3 || randomizer == 4) {
       mover = Random.Range(1, 2); }
    int operators = Random.Range(1, 2);
    if (operators == 1) {
        randomizer -= mover;}
    else{ randomizer += mover;}
    }
 else if (randomizer == 1) {
    int mover = Random.Range(1, 5);
     randomizer += mover; }
}
}
//repeat note agar posisi note tetap sama apabila
nada yang dibunyikan serta velocitynya sama
if (m.data1 == temp_data1 && m.data2==temp_data2)
{randomizer = temp; }

 // ladder note
else if (m.data1 > temp_data1 && m.data1 <=
temp_data1 + byte.Parse("20")){
//berlaku untuk channel melody dan channel rhythm
ketika dimainkan Bersama
if (m.status == mainmelody){
 if (temp < 6){
   randomizer = temp + 1; }
```

```
 //reset
  else if (temp >= 6) {
   randomizer = 4; }
}
else if(m.status == rhythm){
 if(temp<3) {
   randomizer = temp + 1; }
 else if(temp >= 3) {
    randomizer = 1;}
}
//tanpa channel melody dan rhythm
else{
 if (temp < 6) {
    randomizer = temp + 1; }
 //reset
 else if (temp == 6) {
    randomizer = 1; }
}
}
//reverse ladder note
else if (temp_data1 > m.data1 && m.data1 >=
temp_data1 - byte.Parse("20")){
//berlaku untuk channel melody dan channel rhythm
ketika dimainkan Bersama
if (m.status == mainmelody) {
 if (temp >4) {
   randomizer = temp - 1;
 }
//reset
else if (temp == 4) {
  randomizer = 6;}
}
else if (m.status == rhythm){
 if (temp > 1) {
   randomizer = temp - 1; }
 else if (temp == 1){
   randomizer = 3; }}
 else{
   if (temp > 1) {
     randomizer = temp - 1; }
 //reset
 else if (temp == 1) {
     randomizer = 6;
 }}}
//menyimpan data temp untuk proses berikutnya
 temp_data1 = m.data1;
 temp_data2 = m.data2;
 temp = randomizer;
 if (randomizer == 1){ Instantiate(noteA);}
 else if (randomizer == 2 ) {
```

```
  Instantiate(notes);}
else if (randomizer == 3 ) {
  Instantiate(noteD);}
else if (randomizer == 4) {
  Instantiate(noteJ);}
else if (randomizer == 5) {
  Instantiate(noteK);}
else if (randomizer == 6) {
  Instantiate(noteL);}
}
```

- Pemain Menekan *Note* yang akan datang saat permainan berlangsung.

❖ Pemain akan Menekan *Note* yang telah dibuat oleh Midi File Handler. Pada bagian ini akan menjelaskan tentang bagaimana juga implementasi aturan permainan yang telah dijelaskan bab sebelumnya.

➢ Pemain Menekan *Note*, maka yang akan digerakan oleh pemain adalah Aktivator, dimana aktivator akan berjalan dengan fungsi update() pada activator.cs.

```
void Update () {
//apabila menekan Aktivator dan aktif, score
Akan bertambah
if (Input.GetKeyDown (key) && active) {
  GetComponent<Animation>().Play();
  gm.GetComponent<GameManager>().AddCombo();
  Destroy (note);
  Addscore ();
  active = false;
 }
//apabila menekan aktivator namun tidak aktif
Menyebabkan berkurangnya groove gauge dan combo
hilang
else if (Input.GetKeyDown (key) && !active) {
   StartCoroutine(pressed());
}
}
```

➢ Dimana kondisi aktif adalah kondisi dimana *note* sudah menyentuh aktivator, sehingga saatnya untuk ditekan. Kondisi aktif dan non-aktif dapat dijelaskan pada potongan *source code* activator.cs berikut.

```
//fungsi ini adalah dimana sebuah note sudah
Menncapai activator sehingga active menjadi true
```

```
void OnTriggerEnter2D(Collider2D col){
//apabila aktivator terkena note
if (col.gameObject.tag == "Note") {
        note = col.gameObject;
        active = true;
}
//apabila aktivator terkena winNote(winNote
Adalah note yang menandakan permainan menang)
if (col.gameObject.tag == "winNote") {
        Destroy (col.gameObject);
        gm.GetComponent<GameManager>().win();
}
}
//fungsi ini adalah dimana sebuah note sudah
Meninggalkan aktivator sehingga active menjadi
False kembali.
void onTriggerExit2D(Collider2D col){
active= false;
gm <GameManager>().ResetCombo();
}
```

- Setelah pemain menekan yombol, maka sistem akan memberikan suatu keputusan berupa berapa skor pada satu *note* dalam Menekan *Note*, bertambah atau berkurang kah *life gauge*, apakah rantai tetap dilanjutkan. Semua ini keputusan ada pada fungsi-fungsi yang terdapat pada gameManager. Berikut beberapa fungsi yang penting dalam keputusan pada GameManager.cs.

  - AddCombo(): adalah fungsi dimana pemain berhasil menangkap *note* sehingga, combo ditambah, selain itu akan menambah *lifegauge* sebanyak 0.5 persen dari sebelumnya.

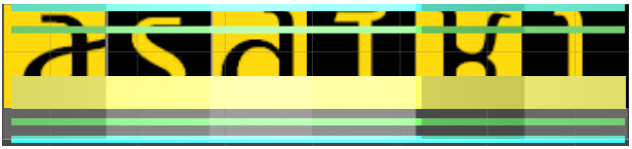
```
public void  AddCombo()
 {
   if(PlayerPrefs.GetInt("GrooveGauge") +0.5<100)
    PlayerPrefs.SetInt("GrooveGauge",
    PlayerPrefs.GetInt("GrooveGauge")+1);
    combo++;
      UpdateGUI ();
      if(combo>PlayerPrefs.GetInt("maxCombo"))
      PlayerPrefs.SetInt("maxCombo" ,combo);
      int maxCombo= PlayerPrefs.GetInt
      ("maxCombo");
      if (accuracylevel == 1)
```

```
        {
           PlayerPrefs.SetInt("Good",
           PlayerPrefs.GetInt("Good") + 1);
          GameObject.Find("Judgement").
          SendMessage("Good");
        }
        else if (accuracylevel == 2 )
        {
            PlayerPrefs.SetInt("Great",
            PlayerPrefs.GetInt("Great") + 1);

        GameObject.Find("Judgement").SendMessage
        ("Great");
        }
        else if(accuracylevel == 3)
        {

        PlayerPrefs.SetInt("Perfect",
        PlayerPrefs.GetInt("Perfect") + 1);

        GameObject.Find("Judgement").SendMessage
        ("Perfect");
        }
     PlayerPrefs.SetInt("notecount",
     PlayerPrefs.GetInt("notecount")+1);
}
```

- AddScore(): adalah fungsi dimana pemain berhasil menangkap *note* sehingga, skor ditambah.

```
public void Addscore()
    {
        int adder;
        if (accuracylevel == 1)
        {
            adder = 50;
        }
        else if (accuracylevel == 2)
        {
            adder=  75;
        }
        else
            adder =  100;
        PlayerPrefs.SetInt("Score",
        PlayerPrefs.GetInt("Score") + adder);
}
```

- Dalam fungsi AddCombo() dan AddScore() terdapat variabel "accuracylevel" dimana *accuracy level* adalah hasil dari pendapatan *judgement*, gambar berikut adalah bagaimana pengimplementasian sistem *judgement*.



Gambar 4.11 bagaimana sistem *Judgement* di implementasikan

Pada **Gambar 4.11** ditunjukkan bahwa pada aktivator memiliki 5 *collider* yang berfungsi sebagai penentu nilai accuracylevel. tiap 5 *collider* memiliki *script* yang akan mengganti accuracylevel tersebut.
Pada *collider* berwarna kuning akan memiliki *script* perfectJudgement.cs, yaitu sebagai berikut.

```
void OnTriggerEnter2D(Collider2D col)
{
        GameManager.accuracylevel = 3;
}
```

Sedangkan pada warna hijau akan memiliki komponen *script* GreatJudgement.cs, yaitu sebagai berikut.

```
void OnTriggerEnter2D(Collider2D col)
{
        GameManager.accuracylevel = 2;
}
```

Sedangkan pada warna biru muda akan memiliki komponen *script* GoodJudgement.cs, yaitu sebagai berikut.

```
void OnTriggerEnter2D(Collider2D col)
{
        GameManager.accuracylevel = 1;
}
```

- ResetCombo(): adalah satu-satunya fungsi dimana pemain tidak berhasil menangkap suatu *note*, menyebabkan *combo* menjadi 0 dan *life gauge* berkurang sebanyak 4%.

```
public void ResetCombo()
 {
        PlayerPrefs.SetInt("GrooveGauge",
        PlayerPrefs.GetInt("GrooveGauge")-4);
        combo = 0;
       UpdateGUI ();
 }
```

- Sistem menampilkan Halaman selanjutnya setelah Game berakhir.
  Ada 2 Halaman yang akan di tampilkan oleh sistem ketika game berakhir, 2 Halaman tersebut meliputi:

  - Halaman Hasil/Result: halaman yang menunjukkan bahwa pemain berhasil untuk bertahan hingga akhir lagu. Fungsi untuk menuju halaman hasil adalah win() dari Gamemanager.cs.

```
public void win()
 {
        PlayerPrefs.SetInt("Start", 0);
        SceneManager.LoadScene (4);
 }
```

    Fungsi win dipanggil apabila aktivator terkena sebuah win*Note* diinstansiasi oleh TestSequencer.cs lalu win*Note* terebut terkena Aktivator.

  - Halaman gagal/"sayang sekali": halaman yang menunjukkan bahwa pemain tidak berhasil untuk bertahan hingga akhir lagu. Fungsi untuk menuju halaman ini adalah fungsi gameover() dari gamemanager.cs.

```
 void GameOver()
 {
        PlayerPrefs.SetInt("Start", 0);
        SceneManager.LoadScene(6);
 }
```

yang akan di panggil oleh fungsi update dari gamemanager.cs sebagai berikut.

```
void Update () {
    if (PlayerPrefs.GetInt("GrooveGauge") <= 0)
      {
         GameOver();
      }}
```

7. **Implementasi Kasus Penggunaan Kembali ke Pemilihan Lagu.**

- Pemain menekan tombol “selanjutnya” (tombol ada pada halaman “hasil” dan halaman “Gagal”).
  tombol onclick akan memanggil fungsi berikut.
  `ButtonFunction.Load*Scene*(2);`
- Sistem Menampilkan Daftar Lagu.
  Setelah mendapatkan panggilan fungsi maka, sistem langsung mengakses fungsi Load*Scene*() script ButtonFunction.cs, dimana a = 2.

```
public void LoadScene(int a)
 {
     if(a == 2|| a== 5)
     {
         emptythesongs();
     }
     SceneManager.LoadScene(a);
 }
 void emptythesongs()
 {
     PlayerPrefs.SetString("songName", "");
     PlayerPrefs.SetString("songBPM", "");
     PlayerPrefs.SetString("songLevel", "");
     PlayerPrefs.SetString("Custom.ini", "");
     PlayerPrefs.SetString("Custom.bytes", "");
     PlayerPrefs.SetString("Custom.wav,"");}
```

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan mengenai rangkaian uji coba dan evaluasi yang dilakukan. Proses pengujian dilakukan menggunakan metode kotak hitam berdasarkan skenario yang telah ditentukan dan pengujian dilakukan dengan survei langsung kepada pengguna.

5.1 Lingkungan Uji Coba

Lingkungan pelaksanaan uji coba meliputi perangkat keras dan perangkat lunak yang akan digunakan pada sistem ini. Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam rangka uji coba perangkat lunak ini dicantumkan pada **Tabel 5.1**.

Tabel 5.1 Lingkungan Uji Coba Perangkat Lunak

Perangkat Keras		Perangkat Lunak	
Prosesor	Memori	Sistem Operasi	Perangkat Pengembang
Intel Core-i5 4210	4 GB RAM	Windows 10 Profesional 64 bit	Unity Engine 2017.2.2f1

5.2 Pengujian kesesuaian *gameplay* dengan aturan permainan

Pengujian ini dilakukan untuk menguji apakah permainan/*gameplay* dapat berjalan sesuai dengan aturan permainan yang sudah ditentukan (lebih tepatnya pada **Subbab 3.3.5**), pada pengujian ini akan membuktikan apakah kondisi

menang atau kalah, sistem rantai, sistem *judgement* dan penilaian serta *life gauge* pada game ini akan dapat berjalan.

5.2.1 Pengujian terhadap *Judgement* dan penilaian dalam *gameplay*

Pengujian ini berfungsi untuk membuktikan apakah sistem *judgement* dapat berjalan sehingga menghasilkan output berupa tulisan apa yang akan dikeluarkan dan berapa nilai yang akan diberikan setiap *note* ditekan. Skenario pada pengujian ini tertera pada **Tabel 5.2**.

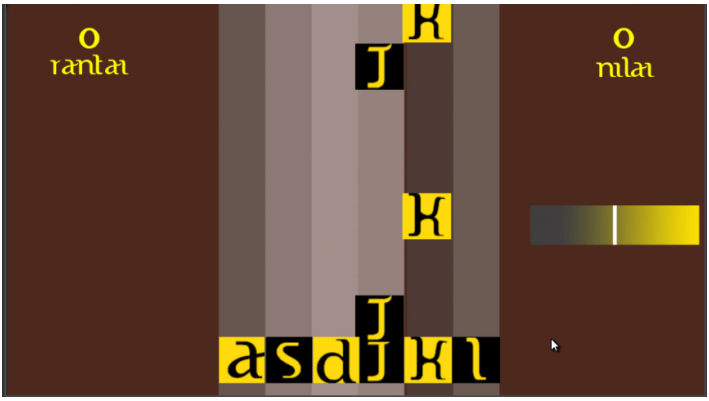
Tabel 5.2 Pengujian *Judgement* dalam Permainan

Kondisi Awal	Game berlangsung dan *note* segera menuju ke aktivator.
Prosedur Pengujian	Pengguna/pemain akan menekan tombol saat *note* sudah menyentuh aktivator setepat mungkin mungkin sehingga menghasilkan *judgement* tertentu.
Hasil yang diharapkan	Sistem akan menampilkan tulisan apa yang dikeluarkan dan skor yang besarnya sesuai dengan akurasi yang di peroleh pengguna/pemain.
Hasil yang diperoleh	Sistem dapat menampilkan tulisan yang sesuai dengan akurasi/*judgement* yang diperoleh pengguna/pemain serta sistem memberikan skor yang sesuai dengan akurasi/*judgement*.
Kesimpulan	Pengujian berhasil.

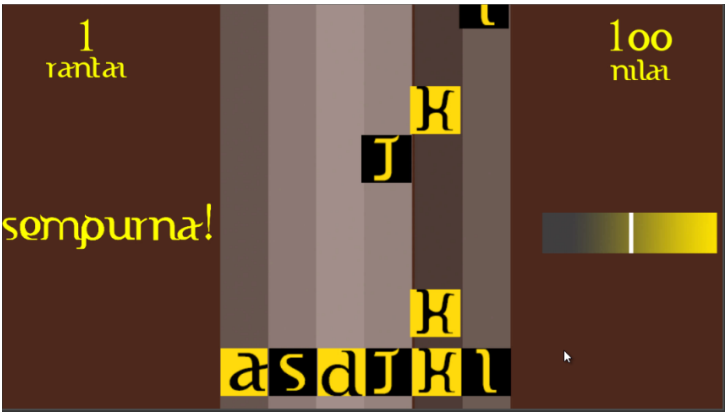
5.2.1.1 Pengujian Kondisi *Judgement* "Sempurna"

Pengujian ini dilakukan dengan menekan *note* setepat mungkin yaitu disaat *note* berada di posisi yang sangat tepat dengan aktivator sehingga menghasilkan kondisi *judgement*

"sempurna" dan nilai akan ditambah sebesar 100. Kondisi awal sebelum *note* ditekan terlihat pada **Gambar 5.1** lalu pada akhirnya pemain menekan *note* sehingga terjadi kondisi *judgement* "sempurna" seperti pada **Gambar 5.2**.



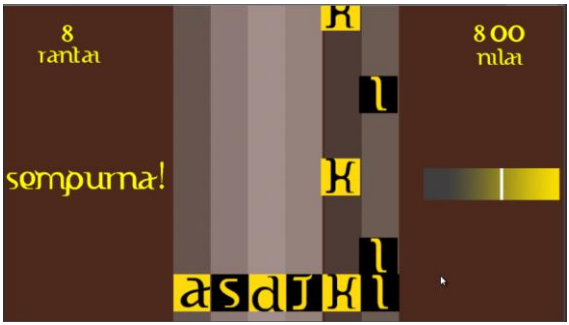
Gambar 5.1 Kondisi sebelum terjadi kondisi *Judgement* "sempurna"



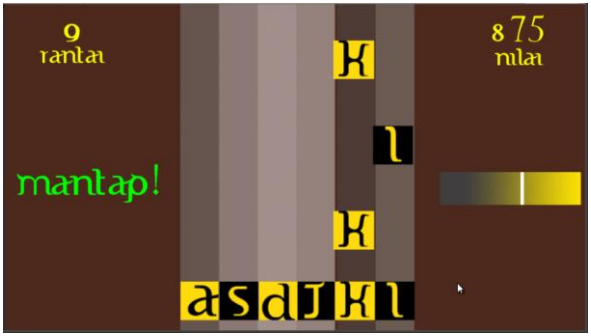
Gambar 5.2 Kondisi setelah *note* menuju aktivator dan ditekan sehingga terjadi kondisi *Judgement* "sempurna"

5.2.1.2 Pengujian Kondisi *Judgement* "mantap"

Pengujian ini dilakukan dengan menekan *note* sehingga memungkinkan terjadi kondisi *judgement* "mantap" dan menghasilkan penambahan nilai sebesar 75, biasanya posisi *note* hampir menutupi aktivator namun masih sedikit bergeser. Kondisi awal sebelum *note* ditekan terlihat pada **Gambar 5.3** lalu pada akhirnya pemain menekan *note* sehingga terjadi kondisi *judgement* "mantap" seperti pada **Gambar 5.4**.



Gambar 5.3 Kondisi sebelum terjadi kondisi *Judgement* "mantap"

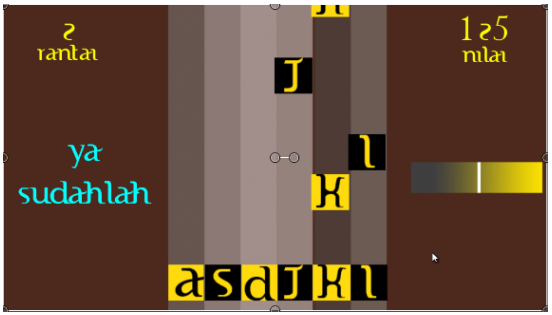


Gambar 5.4 Kondisi setelah *note* menuju aktivator dan ditekan sehingga terjadi kondisi *Judgement* "mantap"

5.2.1.3 Pengujian Kondisi *Judgement* "ya sudahlah"

Pengujian ini dilakukan dengan menekan *note* sehingga memungkinkan terjadi kondisi *judgement* "ya sudahlah" dan menghasilkan penambahan nilai sebesar 50, dimana *note* ditekan pada saat posisi *note* masih belum menutupi aktivator namun setidaknya telah menyentuh aktivator. Kondisi awal sebelum *note* ditekan terlihat pada **Gambar 5.5**, lalu pada akhirnya pemain menekan *note* sehingga terjadi kondisi *judgement* "ya sudahlah" seperti pada **Gambar 5.6**.

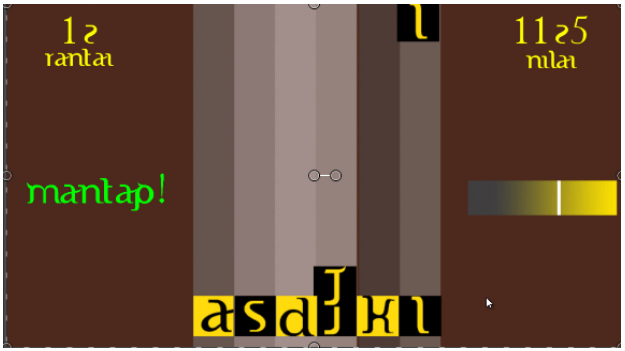
Gambar 5.5 Kondisi sebelum terjadi kondisi *Judgement* "ya sudahlah"



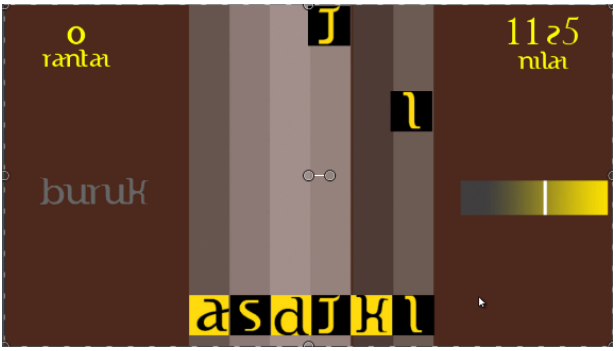
Gambar 5.6 Kondisi setelah *note* menuju aktivator dan ditekan sehingga terjadi kondisi *Judgement* "ya sudahlah"

5.2.1.4 Pengujian Kondisi *Judgement* "buruk"

Pengujian ini dilakukan dengan mengabaikan *note* sehingga memungkinkan terjadi kondisi *judgement* "buruk" dan tidak ada penambahan nilai. Kondisi awal sebelum *note* melewati aktivator terlihat pada **Gambar 5.7** lalu pada akhirnya *note* meninggalkan aktivator sehingga menghasilkan kondisi *judgement* "buruk" seperti pada **Gambar 5.8**.



Gambar 5.7 Kondisi sebelum terjadi kondisi *Judgement* "buruk"



Gambar 5.8 Kondisi setelah *note* menuju aktivator dan ditekan sehingga terjadi kondisi *Judgement* "buruk"

5.2.2 Pengujian terhadap rantai dalam *gameplay*

Pengujian ini berfungsi untuk membuktikan apakah sistem rantai dapat berjalan sehingga menghasilkan output antara menambah rantai setiap *note* berhasil ditekan atau mengulang rantai menjadi 0 apabila *note* lolos atau gagal ditekan. Skenario pada pengujian ini tertera pada **Tabel 5.3**.

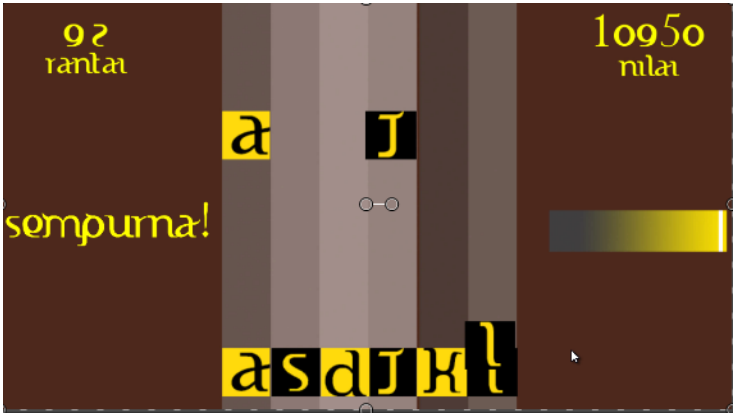
Tabel 5.3 Pengujian Sistem rantai dalam Permainan

Kondisi Awal	Game berlangsung dan *note* segera menuju ke aktivator, rantai udah berada pada nilai tertentu.
Prosedur Pengujian	Pengguna/pemain akan menekan tombol saat *note* sudah menyentuh aktivator sehingga menambah rantai.
Hasil yang diharapkan	Sistem akan menentukan rantai akan bertambah apabila berhasil menekan *note* dan mengulang rantai menjadi 0 apabila *note* lolos atau pemain menekan aktivator ketika *note* tidak berada pada posisi menyentuh aktivator.
Hasil yang diperoleh	Sistem menambah rantai apabila pemain berhasil menekan *note* dan sistem akan mengulang nilai rantai menjadi 0 apabila *note* lolos atau ketika *note* tidak berada pada posisi menyentuh aktivator.
Kesimpulan	Pengujian berhasil.

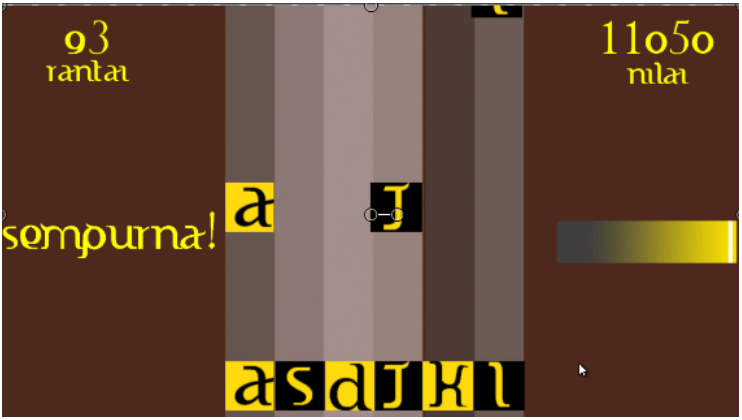
5.2.2.1 Pengujian Penambahan Rantai

Pengujian ini dilakukan dengan menekan *note* ketika *note* telah menyentuh aktivator terlihat pada **Gambar 5.9** lalu pada akhirnya *note* ditangkap oleh aktivator sehingga menambah rantai

sebanyak 1 dari jumlah rantai sebelumnya seperti yang terlihat pada **Gambar 5.10**.



Gambar 5.9 Kondisi Rantai Sebelum Menekan *Note* selanjutnya



Gambar 5.10 Kondisi Rantai Setelah Menekan *Note* dan menambah rantai

5.2.2.2 Pengujian Pengulangan Rantai menjadi 0

Pengujian ini dilakukan dengan mengabaikan *note*, ketika *note* telah berada di bawah aktivator, seperti yang ditunjukan pada **Gambar 5.11** lalu pada akhirnya *note* lolos dan rantai akan direset menjadi 0, seperti yang ditunjukan pada **Gambar 5.12**.

Gambar 5.11 Kondisi Rantai Sebelum *note* lolos dari aktivator

Gambar 5.12 Kondisi Rantai Setelah *note* lolos dari aktivator

5.2.3 Pengujian terhadap *Life Gauge* dalam *gameplay*

Pengujian ini berfungsi untuk membuktikan apakah dalam permainan dapat mempengaruhi *life gauge* ketika *note* ditekan maupun *note* lolos dari aktivator. Skenario pada pengujian ini tertera pada **Tabel 5.4**.

Tabel 5.4 Pengujian *Life Gauge* dalam Permainan

Kondisi Awal	Game berlangsung dan *life gauge* masih berada posisi tengah.
Prosedur Pengujian	Pengguna/pemain akan menekan tombol saat *note* sudah menyentuh aktivator sehingga menggeser *life gauge* ke kanan.
Hasil yang diharapkan	Sistem akan menggeser *life gauge* kekanan ketika pemain/pengguna berhasil menekan *note* dan menggeser *life gauge* ke kiri ketika *note* lolos dari aktivator.
Hasil yang diperoleh	Sistem menggeser *life gauge* kekanan ketika pemain/pengguna berhasil menekan *note* apabila *life gauge* belum berada pada titik maksimal dan menggeser *life gauge* ke kiri ketika *note* lolos dari aktivator.
Kesimpulan	Pengujian berhasil.

5.2.3.1 Pengujian *Life gauge* ketika *note* berhasil ditekan

Pengujian ini dilakukan dengan menekan *note* ketika *note* telah menyentuh aktivator terlihat pada **Gambar 5.13** lalu pada akhirnya *note* ditangkap oleh aktivator sehingga menggeser *life gauge* sedikit kekanan terlihat pada **Gambar 5.14**. apabila *life gauge* telah berada pada titik maksimal maka *life gauge* tidak akan bergeser kekanan kembali, kondisi ini ditunjukkan pada **Gambar 5.15**.

Gambar 5.13 Kondisi *Life Gauge* Sebelum Menekan *Note* selanjutnya

Gambar 5.14 Kondisi *Life Gauge* Setelah menekan beberapa *Note*

Gambar 5.15 Kondisi *Life Gauge* Setelah berada pada titik maksimal

5.2.3.2 Pengujian *Life Gauge* ketika *note* lolos dari aktivator

Pengujian ini dilakukan dengan mengabaikan *note*, ketika *note* telah berada di bawah aktivator **Gambar 5.16** lalu pada akhirnya *note* lolos dan *life gauge* bergeser kekiri **Gambar 5.17.**

Gambar 5.16 Kondisi *Life Gauge* sebelum *Note* lolos dari Aktivator

Gambar 5.17 Kondisi *Life Gauge* setelah *Note* lolos dari Aktivator

5.2.4 Pengujian terhadap kondisi menang dalam *gameplay*

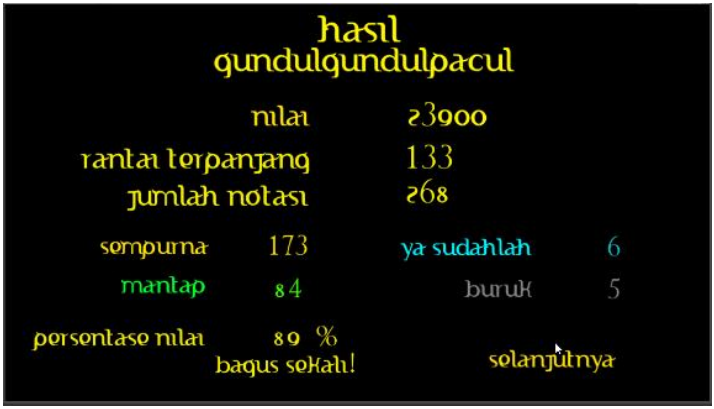
Pengujian ini dilakukan setelah *life gauge* dan aturan permainan lainya dapat berfungsi dengan baik pengujian ini berfungsi untuk membuktikan apakah dalam permainan kondisi menang dapat berfungsi dimana kondisi menang terjadi apabila pemain dapat mempertahankan *life gauge* hingga akhir dari lagu, kondisi menang ditunjukkan dengan menampilkan halaman hasil main dari pemain menunjukkan ringkasan skor, *judgement*, rantai terpanjang dan sebagainya. Skenario pada pengujian ini tertera pada **Tabel 5.5**.

Tabel 5.5 Pengujian kondisi menang dalam Permainan

Kondisi Awal	Lagu sudah mendekati akhir dan *life gauge* masih belum berada pada pojok kiri.
Prosedur Pengujian	Pengguna/pemain akan mempertahankan *life gauge* tersebut dengan tidak membiarkan *note* lolos dari aktivator, hingga lagu berakhir.
Hasil yang diharapkan	Sistem akan menampilkan halaman hasil dari permainan, berupa jumlah nilai, rantai, *note* dan *judgement.*
Hasil yang diperoleh	Sistem menampilkan halaman hasil dari permainan, berupa jumlah nilai, rantai, *note* dan *judgement.*
Kesimpulan	Pengujian berhasil.

Pengujian ini dilakukan dengan mempertahankan *life gauge* hingga akhir lagu, sehingga pemain telah mendapatkan rantai dan nilai yang ditunjukkan **Gambar 5.18** lalu pada akhirnya *note* lolos dan *life gauge* bergeser kekiri **Gambar 5.19.**

Gambar 5.18 Kondisi sebuah lagu telah berakhir



Gambar 5.19 Tampilan Hasil

5.2.5 Pengujian terhadap kondisi kalah dalam *gameplay*

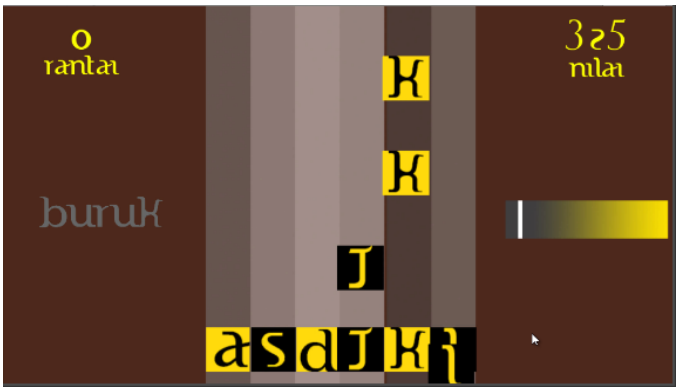
Pengujian ini dilakukan setelah *life gauge* dan aturan permainan lainya dapat berfungsi dengan baik pengujian ini berfungsi untuk membuktikan apakah dalam permainan kondisi kalah dapat berfungsi dimana kondisi kalah terjadi apabila pemain

tidak mempertahankan *life gauge* sehingga *life gauge* mencapai titik minimal, kondisi kalah hanya menunjukkan halaman "sayang sekali", skenario pada pengujian ini ditunjukkan pada **Tabel 5.6**.

Tabel 5.6 Pengujian kondisi kalah dalam Permainan

Kondisi Awal	*Life gauge* sudah mendekati titik minimal.
Prosedur Pengujian	Pengguna/pemain akan mengabaikan *notes* hingga melewati aktivator.
Hasil yang diharapkan	Sistem akan menampilkan halaman hasil "sayang sekali".
Hasil yang diperoleh	Sistem menampilkan halaman hasil "sayang sekali".
Kesimpulan.	Pengujian berhasil.

Pengujian ini dilakukan dengan mengabaikan *note*, sehingga *life gauge* berada pada posisi kritis seperti yang ditunjukkan **Gambar 5.20** lalu pada akhirnya *note* lolos dan *life gauge* Bergeser kekiri **Gambar 5.21**.



Gambar 5.20 Kondisi sebuah Permainan akan mendekati Gagal

Gambar 5.21 Tampilan "Sayang Sekali"

5.2.6 Hasil Uji Fungsionalitas Aturan Permainan

Hasil uji fungsionalitas aturan permainan yang sudah dilakukan dari sistem *judgement* hingga kondisi menang kalah. dapat ditunjukkan bahwa aturan permainan secara sempurna berhasil diimplementasikan. Hasil pengujian fungsionalitas dicantumkan pada **Tabel 5.7**.

Tabel 5.7 Hasil Pengujian Fungsionalitas

No	Hal yang diujikan	Hasil Pengujian
1	*Judgement* dan Penilaian	Berhasil
2	Rantai	Berhasil
3	*Life gauge*	Berhasil
4	Kondisi Kalah	Berhasil
5	Kondisi Kalah	Berhasil

5.3 Uji Midi File Handler

Tujuan dari pengujian ini adalah untuk menguji tiga hal yaitu, konsistensi, kedinamisan serta ketepatan *timing* dari *plugin* Midi File Handler. Yang disebut dengan konsistensi adalah bagaimana Midi File Handler dapat memunculkan banyak *note* dengan jumlah *note* yang memungkinkan sama atau mendekati serta penerapan terhadap variasi *note* apabila sebuah lagu dimainkan secara berulang. Kedinamisan Midi File Handler akan berupa bagaimana sebuah Midi File Handler dapat menghasilkan *notes* yang berbeda pada setiap perubahan lagu midi. Lalu hal terakhir yang di uji adalah ketepatan *timing*, adalah bagaimana sebuah Midi File Handler dapat bekerja dengan kinerja yang baik sehingga *note* yang dibuat akan memiliki *timing* yang sesuai dengan lagu yang dimainkan.

5.3.1 Pengujian Konsistensi Midi File Handler

5.3.1.1 Skenario Pengujian Konsistensi Midi File Handler

Pengujian konsistensi akan melakukan pengulangan pada lagu midi yang sama, sehingga akan terlihat apakah banyak *note* akan setidaknya memiliki selisih yang minim atau sebisa mungkin sama dengan permainan sebelumnya. Berikut langkah pengujiannya:

- Langkah pertama: mengambil 5 file midi internal (terdiri dari file mp3, bytes dan ini) dan 5 file midi eksternal (terdiri dari file wav, bytes dan ini). Isi pada file .ini pada 10 lagu, akan dilampirkan pada **Lampiran 2**.
- Langkah kedua: mengkonfigurasi file midi dengan meng-*edit* file .ini yang mewakili satu set file midi.
- Langkah ketiga: memainkan file midi, setiap file midi akan dimainkan sebanyak 4 kali.
- Langkah keempat : setiap percobaan, mencatat jumlah *notes*.

5.3.1.2 Hasil Pengujian Konsistensi Midi File Handler

Hasil pengujian dapat dilihat pada **Tabel 5.8** dan **Tabel 5.9**.

Tabel 5.8 Hasil Uji Konsistensi Midi File Handler pada Data Internal

Data Internal						
Info lagu			Jumlah *note* pada percobaan			
Nama	BPM	Chan Nel	1	2	3	4
Cingcang Keeling	124	7	289	289	295	288
Ampar Ampar Pisang	140	2	345	351	343	343
Cublak-cublak sueng	92	3	150	150	150	150
Manuk dadali	103	4	481	483	483	482
O inani keke	100	1	96	96	96	96

Tabel 5.9 Hasil Uji Konsistensi Midi File Handler pada Data Eksternal

Data Eksternal						
Info lagu			Jumlah *note* pada percobaan			
Nama	BPM	Chan Nel	1	2	3	4
Cinta Terlarang	140	4,9	129	129	129	129
akumaluta pi mau	140	1, 4	241	245	243	243
Jika Cinta	160	4	47	47	47	47

Data Eksternal						
Info lagu			Jumlah *note* pada percobaan			
Nama	BPM	Chan Nel	1	2	3	4
dia						
Jadi apa yang kuinginkan	142	4	101	101	101	101
Ada apa dengan mu	113	1,4	1508	1507	1511	1510

Dari tabel ini dapat dijelaskan bahwa apabila dalam percobaan pada tiap lagu dari pertama hingga keempat kali masih sama, berarti Midi File Handler sudah bersifat konsisten. Beberapa ada yang berubah, salah satu faktornya adalah double *note* serta waktu *limit note*, terutama pada lagu pertama yaitu cingcangkeling. Namun secara keseluruhan Midi File Handler sudah bersifat relatif konsisten karena perubahan tidak ada yang signifikan.

5.3.2 Pengujian kedinamisan Midi File Handler

5.3.2.1 Skenario Pengujian Kedinamisan Midi File Handler

Pengujian kedinamisan akan melakukan load lagu eksternal yang berubah-ubah, tujuannya adalah agar dapat menunjukkan apakah Midi File Handler akan menghasilkan *note* yang berbeda. Berikut langkah pengujianya:

- Langkah pertama: file midi eksternal (terdiri dari file .wav, file .bytes dan file .ini). Isi pada file .ini pada 5 lagu, akan dilampirkan pada **Lampiran 2**.
- Langkah kedua: mengkonfigurasi file midi dengan meng-*edit* file .ini yang mewakili satu set file midi.

- Langkah ketiga: memainkan file midi secara acak, setiap file midi akan dimainkan sekali saja.
- Langkah keempat : setiap percobaan catat jumlah *notes*.
- Langkah kelima : pada lagu kedua dan selanjutnya beri keterangan apabila susunan *notes* mengalami perubahan dari lagu sebelumnya (contoh: apabila lagu kedua memiliki *notes* yang berbeda dengan lagu pertama maka keterangan pada lagu kedua akan ditulis "ya").

5.3.2.2 Hasil Pengujian Kedinamisan Midi File Handler

Hasil pengujian Kedinamisan Midi File Handler dapat dilihat pada **Tabel 5.10.**

Tabel 5.10 Hasil Uji Kedinamisan Midi File Handler pada Data Eksternal

Eksternal data				
Info lagu			Hasil percobaan	
Nama	BPM	Channel	Jumlah *Notes*	Terjadi Perubahan?
Jika Cinta Dia	160	4	47	(awal lagu)
Ada apa Denganmu	113	1, 4	1508	ya
Jadi apa yang kuinginkan	142	4	101	Ya
Cinta Terlarang	142	4,9	129	Ya
akumaluta pimau	140	1,4	243	Ya

Dari tabel ini dapat dijelaskan bahwa setiap perubahan lagu, juga terjadi perubahan jumlah *notes* yang berarti Midi File Handler ini bersifat dinamis karena dapat menyesuaikan *notes* dengan konfigurasi lagu dan banyaknya *midi event* pada lagu tersebut.

5.3.3 Pengujian Ketepatan *Timing* Midi File Handler

5.3.3.1 Skenario Pengujian Ketepatan *Timing* Midi File Handler

Pengujian ketepatan *timing* akan melakukan hal yang sama pada pengujian konsistensi sebelumnya (pada **Subbab 5.3.1)**, berikut langkah pengujianya:

- Langkah pertama: mengambil 5 file midi internal (terdiri dari file .mp3, file .bytes dan file .ini) dan 5 file midi eksternal (terdiri dari file .wav, file .bytes dan file .ini). Isi pada file .ini pada 10 lagu, akan dilampirkan pada **Lampiran 2**.
- Langkah kedua: mengkonfigurasi file midi dengan meng*edit* file .ini yang mewakili satu set file midi.
- Langkah ketiga: memainkan file midi, setiap file midi akan dimainkan sebanyak 4 kali.
- Langkah keempat : catat setiap pengujian apakah *timing* antara *note* dengan lagu sudah tepat pada konfigurasi *time delay* dan bpm yang sudah ditentukan.

Hasil pengujian ketepatan *timing* Midi File Handler Hasil Pengujian Dapat dilihat pada **Tabel 5.11** dan **Tabel 5.12**. Pada pengujian ini ditunjukan bahwa apabila konfigurasi lagu midi sudah benar terutama pada BPM dan *time delay*nya maka *timing* akan selalu tepat walaupun sudah dilakukan perulangan tiap lagunya, baik Data Internal maupun Data Eksternal tidak ada perubahan terhadap performa, sehingga tidak akan mempengaruhi *timing* dalam permainan.

Tabel 5.11 Hasil Uji Ketepatan *Timing* Midi File Handler pada Data internal

Data Internal						
Info lagu			Apakah *Timing* sudah tepat?			
Nama	BPM	Time Delay	1	2	3	4
Cingcang Keeling	124	1,5 detik	Ya	Ya	Ya	Ya
Ampar Ampar Pisang	140	1 detik	Ya	Ya	Ya	Ya
Cublak-cublak sueng	92	0.4 detik	Ya	Ya	Ya	Ya
Manuk dadali	103	2.2 detik	Ya	Ya	Ya	Ya
O inani keke	100	1 detik	Ya	Ya	Ya	Ya

Tabel 5.12 Hasil Uji Ketepatan *Timing* Midi File Handler pada Data eksternal

Eksternal data						
Info lagu			Apakah *Timing* sudah tepat?			
Nama	BPM	Time Delay	1	2	3	4
Cinta Terlarang	140	2,3 detik	Ya	Ya	Ya	Ya
akumalutapi mau	140	2,3 detik	Ya	Ya	Ya	Ya
Jika Cinta dia	160	2,3 detik	Ya	Ya	Ya	Ya
Jadi apa yang kuinginkan	142	2.4 detik	Ya	Ya	Ya	Ya

Eksternal data						
Info lagu			Apakah *Timing* sudah tepat?			
Nama	BPM	Time Delay	1	2	3	4
Ada apa dengan mu	113	2,3 detik	Ya	Ya	Ya	Ya

5.4 Pengujian Pengguna

Pengujian pada perangkat lunak yang dibangun tidak hanya dilakukan pada fungsionalitas yang dimiliki, tetapi juga pada pengguna untuk mencoba secara langsung. Pengujian ini berfungsi sebagai pengujian subjektif yang bertujuan untuk mengetahui tingkat keberhasilan aplikasi yang dibangun dari sisi pengguna. Hal ini dapat dicapai dengan meminta penilaian dan tanggapan dari pengguna terhadap sejumlah aspek perangkat lunak yang ada.

5.4.1 Skenario Uji Coba Pengguna

Dalam melakukan pengujian perangkat lunak, penguji diminta mencoba menggunakan perangkat lunak untuk mencoba semua fungsionalitas dan fitur yang ada. Pengujian aplikasi oleh pengguna dilakukan dengan penjelasan singkat kepada penguji tentang bagaimana cara bermain dan apa saja yang perlu di operasikan. Setelah informasi tersampaikan, pengguna kemudian diarahkan untuk langsung mencoba aplikasi dengan spesifikasi lingkungan yang sama dengan yang telah diuraikan pada uji coba fungsionalitas.

Jumlah pengguna yang terlibat dalam pengujian perangkat lunak sebanyak enam orang pada kalangan anak muda yang setidaknya mengerti sistem *Rhythm Game*. Dalam melakukan pengujian, pengguna melakukan percobaan lebih dari satu lagu bebas dipilih oleh pengguna sendiri.

Dalam memberikan penilaian dan tanggapan, penguji diberikan formulir pengujian perangkat lunak. Formulir pengujian perangkat lunak ini memiliki beberapa aspek penilaian dan pada bagian akhir terdapat saran untuk perbaikan fitur.

5.4.2 Daftar Penguji Perangkat Lunak

Pada subbab ini ditunjukkan daftar pengguna yang bertindak sebagai penguji coba aplikasi yang dibangun. Daftar nama penguji aplikasi ini dapat dilihat pada **Tabel 5.13**.

Tabel 5.13 Daftar Nama Penguji Coba Aplikasi

No	Nama	Pekerjaan
1	Rifqi Maula Iqbal	Alumni Mahasiswa Teknik Informatika ITS
2	Monica Indah hapsari	Mahasiswa Departemen Informatika ITS
3	Arfian Fidiantoro	Mahasiswa Departemen Informatika ITS
4	Dzulkarnain	Mahasiswa Departemen Informatika ITS
5	Richardo Tiono	Alumni Mahasiswa Sistem Informasi ITS
6	Joshua Kevin	Mahasiswa Departemen Informatika ITS

5.4.3 Hasil Uji Coba Pengguna

Uji coba yang dilakukan terhadap beberapa pengguna memiliki beberapa aspek yang dipisahkan berdasarkan antarmuka dan fungsionalitas yang dimiliki. Sistem penilaian didasarkan pada skala penghitungan satu sampai lima di mana skala satu menunjukkan nilai terendah dan skala lima menunjukkan skala tertinggi. Penilaian akhir kemudian dilakukan dengan menghitung berapa banyak penguji yang memilih suatu skala tertentu dan kemudian dicari nilai rata-ratanya. Hasil uji coba dipaparkan secara lengkap dengan disertai tabel yang dapat dilihat pada tiap subbab berikut.

5.4.3.1 Hasil Penilaian terhadap Antarmuka

Penilaian antarmuka difokuskan pada penilaian pengguna terhadap kemudahan penggunaan antarmuka dan sifat-sifat lain yang perlu dimiliki. Hasil penilaian pengguna terhadap antarmuka aplikasi dapat dilihat pada **Tabel 5.14**.

Tabel 5.14 Penilaian terhadap Antarmuka

No.	Antarmuka	Penilaian					Rata-Rata
		1	2	3	4	5	
1	Kemudahan Penggunaan	0	1	3	1	1	3,33
2	Responsifitas UI	0	1	1	4	0	3,5
3	Keindahan Tampilan	1	0	1	4	0	3,33
4	Kesesuaian tema	0	0	1	2	3	4,33
5	Kelengkpan UI	0	2	2	1	1	3,16
Nilai Akhir							**3,53**

5.4.3.2 Hasil Penilaian terhadap Performa Sistem

Penilaian performa sistem difokuskan pada penilaian pengguna terhadap kemampuan aplikasi dalam menghasilkan performa dari interaksi pengguna. Penilaian ini juga ditujukkan untuk mendapatkan tingkat kecepatan dan kelancaran sistem atas interaksi yang dibuat oleh pengguna. Hasil penilaian performa sistem dapat dilihat pada **Tabel 5.15**.

Tabel 5.15 Penilaian terhadap Performa Sistem

No.	Antarmuka	Penilaian					Rata-Rata
		1	2	3	4	5	
1	Kelancaran kinerja aplikasi	0	0	0	4	2	4,33
2	Kesesuaian *timing Note* dengan Lagu	0	0	0	3	3	4,5
3	Kesesuaian input dan output	0	0	1	1	4	4,5
Nilai Akhir							**4,44**

5.4.3.3 Hasil Penilaian terhadap Konten

Penilaian konten hanya fokus pada konten yang disediakan dalam aplikasi, tentang kepantasan konten untuk di pakai dan unsur ketertarikan bagi pengguna. Terdapat hasil penilaian terhadap konten dapat dilihat pada **Tabel 5.16**.

Tabel 5.16 Penilaian Terhadap Konten

No.	Antarmuka	Penilaian					Rata-Rata
		1	2	3	4	5	
1	Kepantasan/Appropriasi Konten	0	0	0	3	3	4,5
2	Ketertarikan Terhadap konten	0	1	3	3	0	3,66
						Nilai Akhir	4,08

5.4.3.4 Hasil Penilaian terhadap Desain Game

Penilaian ini berfokus pada desain game, yaitu bagaimana menyeimbangkan game antara susah dan mudah, sehingga pemain tetap tertarik dalam memainkan game tersebut. Hasil penilaian terhadap desain game dapat dilihat pada **Tabel 5.17.**

Tabel 5.17 Penilaian Terhadap Desain Game

No.	Antarmuka	Penilaian					Rata-Rata
		1	2	3	4	5	
1	Variasi tingkat kesulitan	0	0	2	2	2	4
2	Keseimbangan kondisi mnenang/kalah	0	1	2	2	1	3,5
3	Kesesuaian input dan output	0	0	2	2	2	4
						Nilai Akhir	3,83

5.4.4 Hasil Pengujian Pengguna

Evaluasi pengujian pengguna dilakukan dengan menampilkan data rekapitulasi perangkat lunak yang telah dipaparkan. Dalam hal ini, rekapitulasi disusun dalam bentuk tabel yang dapat dilihat pada **Tabel 5.10**. Dari data diketahui bahwa aplikasi hampir memenuhi unsur yang seharusnya seperti perancangan dimana nilai persentase memiliki lebih dari 80%. Arti dari nilai yang memiliki lebih dari 80% ini merupakan penilaian yang sangat baik. Permainan masih belum dinilai sangat baik oleh pengguna, karena masih memiliki aspek yang kurang yaitu pada aspek tampilan serta desain game yang terlalu sulit sebelumnya. Namun aplikasi ini telah unggul pada performa. Dan keseluruhan nilai dari aplikasi ini adalah 79,4% yang berarti masih berada di satas rata-rata.

Tabel 5.18 Rekapitulasi Hasil Uji Coba Pengguna

No.	Nama Pengujian		Rata-Rata	Nilai	Nilai (%)
1.	Penilaian Antarmuka	Kemudahan Penggunaan	3,33	**3,53**	**71**
		Responsitifas UI	3,5		
		Keindahan Tampilan	3,33		
		Kesesuaian tema	4,33		
		Kelengkapan UI	3,16		
2	Performa Sistem permainan	Kelancaran Kinerja Aplikasi	4,33	**4,44**	**88**
		Kesesuaian *timing note* pada lagu	4,5		
		Kesesuaian Input dan Output	4,5		
3	Penilaian Konten	Kepantasan/Appropriasi Konten	4,5	**4,08**	**81**
		Ketertarikan terhadap konten	3,66		
4	Penilaian Desain Game	Variasi Tingkat Kesulitan	4	3,83	**76**
		Keseimbangan kondisi menang/kalah	3,5		
		Kesesuaian input dan output	4		
Rata Rata					**79,4**

(Halaman Ini Sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan perangkat lunak dan hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan perangkat lunak lebih lanjut.

6.1. Kesimpulan

Dalam proses pengerjaan tugas akhir mulai dari tahap analisis, desain, implementasi, hingga pengujian didapatkan kesimpulan sebagai berikut:

1. Berdasarkan uji kesesuaian gameplay dengan aturan permainan sistem pada **Subbab 5.2** *Judgement* dan penilaian sudah dapat berjalan dengan baik dimana nilai yang didapat oleh pemain akan selalu sesuai dengan akurasi yang didapat. Sistem rantai juga dapat berjalan dengan baik sehingga *note* dapat mempengaruhi jumlah rantai tersebut antara bertambah ataupun kembali ke 0, *life gauge* juga dapat bergeser kekiri apabila *note* lolos dan kekanan apabila *note* berhasil ditangkap sehingga dapat memutuskan apakah seorang pemain akan gagal maupun berhasil hingga akhir pada lagu.
2. Berdasarkan uji konsistensi Midi File Handler berjalan relatif konsisten saat 4 kali pengujian, 5 dari 10 lagu yang sudah di uji pada **Subbab 5.3.1**, sama sekali tidak terjadi perubahan pada jumlah *note*, sedangkan 5 dari 10 lainnya terjadi perubahan namun selisihnya tidak pernah lebih dari 10 *note*, faktor dari kejadian tersebut adalah *double note* yang diterapkan dalam lagu dan waktu untuk *limit note* yang terkadang berubah-ubah sedikit namun walaupun demikian perubahan jumlah *note* tidak pernah dalam selisih yang signifikan, berarti Midi File Handler ini berjalan relatif baik.

3. Dalam pengujian kedinamisan Midi File Handler pada **Subbab 5.3.2** dapat disimpulkan bahwa Midi File Handler telah berjalan dengan dinamis dimana lagu yang selalu berubah jumlah *note* pada lagupun berubah dikarenakan *midi event* pada midi juga berubah.
4. Dalam pengujian *timing* Midi File Handler pada **Subbab 5.3.3** dapat disimpulan bahwa *timing* pada Midi File Handler akan bergantung pada BPM dan *time delay* pada masing - masing lagu apabila tepat, maka *timing* akan selalu tepat walaupun lagu tersebut diulang.
5. Pada uji pengguna didapatkan persentase kepuasan pengguna terhadap aplikasi adakah 79,4 % dimana yang menjadi unggul adalah pada aspek performa yaitu 88 % terutama dikarenakan Midi File Handler yang dapat berjalan lancar tanpa membuat komputer menjadi berat. Sedangkan yang masih kurang bagi pengguna adalah pada aspek antarmuka dikarenakan kelengkapan UI kurang lengkap dan keindahan tampilan juga masih kurang.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang, berdasarkan pada hasil perancangan, implementasi dan uji coba yang telah dilakukan.

1. Pada *interface* menu terutama pemilihan lagu dan sebagainya seharusnya diberi keterangan sehingga penguna akan mengerti cara untuk memilih dan membuka lagu untuk bermain game tersebut.
2. Menambah fitur *pause* saat game dimulai.
3. Menambah animasi background pada saat permainan dimulai.

.

DAFTAR PUSTAKA

[1] Wikipedia, "Rhythm Game" Wikipedia, 2017. [Online]. Available: https://en.wikipedia.org/wiki/Rhythm _game [Accessed 11 01 2017].

[2] Unity Technologies, "Object.Instantiate," Unity Technologies, 2017. [Online]. Available: https://docs.Unity3d.com/ScriptReference/Object.Instantiate.html.
[Accessed 17 10 2017].

[3] Unity Technologies, "UnityWebRequest" Unity Technologies, 2017. [Online]. Available: https://docs.Unity3d.com/ScriptReference/Networking.UnityWebRequest.html
[Accessed 11 12 2017].

[4] Unity Technologies, "UnityWebRequestMultimedia," Unity Technologies, 2017. [Online]. Available: https://docs.Unity3d.com/ScriptReference/Networking.UnityWebRequestMultimedia.html
[Accessed 11 12 2017].

[5] Unity Technologies, "MonoBehaviour.StartCoroutine," Unity Technologies, 2017. [Online]. Available: https://docs.Unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html
[Accessed 29 03 2017].

[6] Unity Technologies, "Collider," Unity Technologies, 2017. [Online]. Available: https://docs.Unity3d.com/ScriptReference/Collider.html
[Accessed 13 09 2017].

[7] Unity Technologies, "PlayerPrefs." Unity Technologies, 2017. [Online]. Available: https://docs.Unity3d.com/ScriptReference/PlayerPrefs.html
[Accessed 13 09 2017].

[8] Microsoft, "File.ReadAllBytes.Method," Microsoft, 2010. [Online]. Available: https://msdn.microsoft.com/en-us/library/system.io.file.readallbytes(v=vs.100).aspx [Accessed 11 12 2017].

[9] Wikipedia, "General Midi" Wikiepedia, 2017. [Online]. Available: https://en.wikipedia.org/wiki/General_MIDI [Accessed 29 11 2017].

[10] Anonymous "*Midi event Table*," Anonymous, 2017. [Online].Available:http://www.onicos.com/staff/iz/formats/midi-event.html . [Accessed 29 03 2017].

LAMPIRAN 1

Berikut *source code* versi lengkap pada komponen yang digunakan di masing masing kasus penggunaan yang dapat dilampirkan

1. TestSquencer.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using SmfLite;
using UnityEngine.SceneManagement;
using System.IO;
using System.Linq;
using UnityEngine.Networking;
using System.Text;

public class TestSequencer : MonoBehaviour
{
    public TextAsset sourceFile;
    public AudioClip clip;

    public GameObject noteA, notes, noteD, noteJ, noteK, noteL, spawnSpot;
    bool disableA = false, disableS = false, disableD = false, disableJ = false,
disableK = false, disableL = false;
    MidiFileContainer song;
    MidiTrackSequencer sequencer;

    public float bpm;
    public float notelimittime;
    public bool played= false;
    byte temp_data1 = 0, temp_data2 = 0;
    int notecount;
    float timedelay;
    bool enable;
    byte  curnotemax, curnotemin;
    byte curvelomin, curvelomax;

    List<byte> curstat,curmelodystat,currhythmstat;
    int temp;
    string sourceAlternative;
    bool TemporaryLimit;
    Scene currentScene;
    byte[] custom;
    string sceneName;
```

```
    string filepath;
    void ResetAndPlay ()
    {
        StartCoroutine(DelayAudio());
        sequencer = new MidiTrackSequencer (song.tracks [0], song.division,
bpm);
        ApplyMessages (sequencer.Start ());

    }
    IEnumerator Start ()
    {
    PlayerPrefs.SetInt("noteount", 0);
    noteA = Resources.Load("note-A") as GameObject;
        notes = Resources.Load("note-S") as GameObject;
        noteD = Resources.Load("note-D") as GameObject;
        noteJ = Resources.Load("note-J") as GameObject;
        noteK = Resources.Load("note-K") as GameObject;
        noteL = Resources.Load("note-L") as GameObject;
        played = false;
        temp = 0;
        string Name = PlayerPrefs.GetString("songName");
        if (PlayerPrefs.GetString("Custom.ini") != "")
        {
        filepath = PlayerPrefs.GetString("Custom.ini");
        }
        else filepath = Application.dataPath + "/Resources/MidiConfig/" +
Name + ".ini";
    if (PlayerPrefs.GetString("Custom.wav") != "")
    {
        string CustomAudio = PlayerPrefs.GetString("Custom.wav");
        UnityWebRequest www =
UnityWebRequestMultimedia.GetAudioClip("file://" + CustomAudio,
AudioType.WAV);
        yield return www.Send();

        AudioClip clipper = DownloadHandlerAudioClip.GetContent(www);
        GetComponent<AudioSource>().clip = clipper;
    }
    else
        GetComponent<AudioSource>().clip = Resources.Load("MidiAudio/"
+ Name) as AudioClip;
        if (PlayerPrefs.GetString("Custom.bytes") != "")
        {
            sourceAlternative = PlayerPrefs.GetString("Custom.bytes");
            custom = File.ReadAllBytes(sourceAlternative);
        }
        else {
            sourceFile = Resources.Load("Midi Notes/" + Name) as TextAsset;
        }
```

```
    string config = File.ReadAllLines(filepath).First();
    string[] entries = config.Split('|');
    //inisialisasi data
    if (entries[0] == Name)
    {
        bpm = float.Parse(entries[1]);
        timedelay = float.Parse(entries[2]);
        Debug.Log(bpm + timedelay);
        string[] channels = entries[3].Split(',');
        int i = 0;
        curstat = new List<byte>(new byte[] { });
        curmelodystat = new List<byte>(new byte[] { });
        currhythmstat = new List<byte>(new byte[] { });
          foreach (var channel in channels)
        {


            curstat.Add(byte.Parse(channels[i]));
            Debug.Log(byte.Parse(channels[i]));
            i++;
        }
        curvelomin = byte.Parse(entries[4]);
        notelimittime = float.Parse(entries[6]);
        //jika ada dua channel yang dimainkan bersamaan
        if (entries[7] !="none" && entries[8] != "none")
        {
        i = 0;
        Debug.Log(entries[7]);
        string[] melodychannels = entries[7].Split(',');
        foreach (var melodychannel in melodychannels)
        {
            curmelodystat.Add(byte.Parse(melodychannels[i]));
            i++;
        }
        i = 0;
        string[] rhythmchannels = entries[8].Split(',');
        foreach(var rhythmchannel in rhythmchannels)
        {
            currhythmstat.Add(byte.Parse(rhythmchannels[i]));
            i++;
        }
    }
    }
if (custom != null)
{
    song = MidiFileLoader.Load(custom);
}
else
{ song = MidiFileLoader.Load(sourceFile.bytes); }
```

```
    ResetAndPlay();
    noteount = 0;

}
//untuk menunda wav maupun mp3
IEnumerator DelayAudio()
    {
        yield return new WaitForSeconds(timedelay);

        GetComponent<AudioSource>().Play();
    Debug.Log("played");
    played = true;
    }
    // waktu disable note
    IEnumerator limitNote()
    {
        TemporaryLimit = true;
        yield return new WaitForSeconds(notelimittime);
        TemporaryLimit = false;
    }
    void Update ()
    {
     if (sequencer != null && sequencer.Playing) {
            ApplyMessages (sequencer.Advance (Time.deltaTime));
        }
        if (!GetComponent<AudioSource>().isPlaying && played == true)
        {
            played = false;
            Instantiate(Resources.Load("WinNote"));
        }
    }
    void ApplyMessages (List<MidiEvent> messages)
    {
    Debug.Log(messages);
        if (messages != null) {

            foreach (var m in messages) {

            Debug.Log(m.status);
                if ( curstat.Contains(m.status) && m.data2>=curvelomin)
                {

                //Apabila masih pada limit
                if (TemporaryLimit)
                    {
                        return;
                    }

                    //Arbitrary note diawal
```

```
int randomizer= Random.Range(1, 7);
if (curmelodystat!=null && currhythmstat != null)
{
   if (curmelodystat.Contains(m.status))
   {
      randomizer = Random.Range(4, 7);
   }
   else if (currhythmstat.Contains(m.status))
   {
      randomizer = Random.Range(1, 4);
   }
}
//memulai proses Waktu Limit note
if (notelimittime > 0)
{
   StartCoroutine(limitNote());
}
```

//repeat *note* agar posisi *note* tetap sama apabila nada yang dibunyikan serta *velocity*nya sama

```
if (m.data1 == temp_data1 && m.data2 == temp_data2)
{
   randomizer = temp;
}
```

//agar tidak ada *note* yang tabrakan jika ada *note* double/ *note* yang datang bersamaan

```
if (randomizer == temp)
   {
      if(curmelodystat.Contains(m.status))
      {
         if (randomizer == 6)
         {
           int mover = Random.Range(1, 2);
            randomizer -= mover;
         }
        else if (randomizer == 5 )
        {
           int mover = 1;
           int operators = Random.Range(1, 2);
           if (operators == 1)
           {
              randomizer -= mover;
           }
           else
           {
              randomizer += mover;
           }

        }
```

```
    else if (randomizer == 4)
    {
        int mover = Random.Range(1, 2);
        randomizer += mover;
    }
}
    else if (currhythmstat.Contains(m.status))
    {

        if (randomizer == 3)
        {
            int mover = Random.Range(1, 2);
            randomizer -= mover;
        }
        else if (randomizer == 2)
        {
            int mover = 1;
            int operators = Random.Range(1, 2);
            if (operators == 1)
            {
                randomizer -= mover;
            }
            else
            {
                randomizer += mover;
            }

        }
        else if (randomizer == 1)
        {
            int mover = Random.Range(1, 2);
            randomizer += mover;
        }
    }
    else{
    if (randomizer == 6)
    {
        int mover = Random.Range(1, 5);
        randomizer -= mover;
    }
    else if (randomizer <= 5 && randomizer >= 2)
    {
        int mover = 0;
        if (randomizer == 5 || randomizer == 2)
        {
            mover = 1;

        }
        if (randomizer == 3 || randomizer == 4)
```

```
                {
                    mover = Random.Range(1, 2);
                }
                int operators = Random.Range(1, 2);
                if (operators == 1)
                {
                    randomizer -= mover;
                }
                else
                {
                    randomizer += mover;
                }

            }
            else if (randomizer == 1)
            {
                int mover = Random.Range(1, 5);
                randomizer += mover;
            }
        }
        }


        // ladder note
        else if (m.data1 > temp_data1 && m.data1 <= temp_data1 +
byte.Parse("20"))
        {
        //berlaku untuk channel melody dan channel rhythm ketika
dimainkan bersama
        if (curmelodystat.Contains(m.status))
        {
            if (temp < 6)
            {
                Debug.Log(randomizer + "masuk");
                randomizer = temp + 1;
                Debug.Log("Ladder Up");
            }
            //reset
            else if (temp >= 6)
            {
                randomizer = 4;
            }
        }
        else if(currhythmstat.Contains(m.status))
        {
            if(temp<3)
            {
                randomizer = temp + 1;
            }
```

```
                    else if(temp >= 3)
                    {
                        randomizer = 1;
                    }
                }
                    else
                    {
                        if (temp < 6)
                        {
                            Debug.Log(randomizer + "masuk");
                            randomizer = temp + 1;
                            Debug.Log("Ladder Up");
                        }
                    //reset
                    else if (temp == 6)
                    {
                        randomizer = 1;
                    }
                }
            }
            //reverse ladder note
            else if (temp_data1 > m.data1 && m.data1 >= temp_data1 -
byte.Parse("20"))
            {
                //berlaku untuk channel melody dan channel rhythm ketika
dimainkan bersama
                if (curmelodystat.Contains(m.status))
                {
                    if (temp >4)
                    {
                        Debug.Log(randomizer + "masuk");
                        randomizer = temp - 1;
                        Debug.Log("Ladder Up");
                    }
                    //reset
                    else if (temp == 4)
                    {
                        randomizer = 6;
                    }
                }
                else if (currhythmstat.Contains(m.status))
                {
                    if (temp > 1)
                    {
                        randomizer = temp - 1;
                    }
                    else if (temp == 1)
                    {
                        randomizer = 3;
```

```
            }
        }
        else
        {
            if (temp > 1)
            {
                randomizer = temp - 1;
                Debug.Log("Ladder Down");
            }
            //reset
            else if (temp == 1)
            {
                randomizer = 6;
            }
        }
    }
        temp_data1 = m.data1;
        temp_data2 = m.data2;
        temp = randomizer;
        Debug.Log(randomizer);


    if (randomizer == 1 )
        {
          Instantiate(noteA);
        }
        else if (randomizer == 2 )
        {
            Instantiate(notes);
        }
        else if (randomizer == 3 )
        {
            Instantiate(noteD);
        }
        else if (randomizer == 4)
        {
            Instantiate(noteJ);
        }
        else if (randomizer == 5)
        {
            Instantiate(noteK);
        }
        else if (randomizer == 6)
        {
            Instantiate(noteL);
        }
    }
  }
}
```

```
        }
    }
```

2. Activator.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class activator : MonoBehaviour {
  SpriteRenderer Tr;
  public KeyCode key;
  bool active = false;
  GameObject note, gm;
  // Use this for initialization
  void Awake () {
     Tr = GetComponent<SpriteRenderer> ();
  }
  void Start()
  {
     gm = GameObject.Find ("GameManager");
  }
// Update is called once per frame
void Update () {
     if (Input.GetKeyDown (key) && active) {
        GetComponent<Animation>().Play();
        gm.GetComponent<GameManager>().AddCombo();
        Destroy (note);
        gm.GetComponent<GameManager>().Addscore ();
       active = false;
     }
     else if (Input.GetKeyDown (key) && !active) {
       StartCoroutine(pressed());
       gm.GetComponent<GameManager>().ResetCombo();
      }

   }
   void OnTriggerEnter2D(Collider2D col){
     if (col.gameObject.tag == "Note") {
       note = col.gameObject;
       active = true;}
     if (col.gameObject.tag == "winNote") {
       Destroy (col.gameObject);
       gm.GetComponent<GameManager>().win();
     }
   }
    void onTriggerExit2D(Collider2D col){
      active= false;
      gm.GetComponent<GameManager>().ResetCombo();
     }
```

```
    IEnumerator pressed()
    {
        Color old = Tr.color;
        Tr.color = new Color (0, 0, 0);
        yield return new WaitForSeconds (0.0f);
        Tr.color = old;
    }
}
```

3. GameManager.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.SceneManagement;
public class GameManager : MonoBehaviour {
int combo = 0;
public static int accuracylevel = 0;

// Use this for initialization
void Start () {
 //KosongKan seluruhnya
    PlayerPrefs.SetInt ("Score", 0);
    PlayerPrefs.SetInt ("GrooveGauge", 50);
    PlayerPrefs.SetInt ("Combo", 0 );
    PlayerPrefs.SetInt ("maxCombo", 0 );
    PlayerPrefs.SetInt ("Perfect", 0);
    PlayerPrefs.SetInt("Great", 0);
    PlayerPrefs.SetInt("Good", 0);
    PlayerPrefs.SetInt ("Miss", 0);
    PlayerPrefs.SetInt("Start", 1);
    PlayerPrefs.SetInt("noteount", 0);}
// Update is called once per frame
void Update () {
    if (PlayerPrefs.GetInt("GrooveGauge") <= 0)
    {
        GameOver();
    }
}
void OnTriggerEnter2D(Collider2D col)
{
    PlayerPrefs.SetInt("noteount", PlayerPrefs.GetInt("noteount") + 1);
    PlayerPrefs.SetInt("Miss", PlayerPrefs.GetInt("Miss") + 1);
    GameObject.Find("Judgement").SendMessage("Miss");
    ResetCombo();
    Destroy (col.gameObject);
}
```

```
public void  AddCombo()
{
    if(PlayerPrefs.GetInt("GrooveGauge") +0.5<100)
      PlayerPrefs.SetInt("GrooveGauge",
     PlayerPrefs.GetInt("GrooveGauge")+1);
       combo++;
     UpdateGUI ();
      if (combo > PlayerPrefs.GetInt ("maxCombo"))
          PlayerPrefs.SetInt ("maxCombo",combo);
      int maxCombo = PlayerPrefs.GetInt("maxCombo");
      if (maxCombo > PlayerPrefs.GetInt ("BestMaxCombo"))
           PlayerPrefs.SetInt ("BestMaxCombo", maxCombo);
     if (accuracylevel == 1)
    {
          PlayerPrefs.SetInt("Good", PlayerPrefs.GetInt("Good") + 1);
          GameObject.Find("Judgement").SendMessage("Good");
    }
    else if (accuracylevel == 2 )
   {
          PlayerPrefs.SetInt("Great", PlayerPrefs.GetInt("Great") + 1);
          GameObject.Find("Judgement").SendMessage("Great");
    }
    else if(accuracylevel == 3)
   {
      PlayerPrefs.SetInt("Perfect", PlayerPrefs.GetInt("Perfect") + 1);
      GameObject.Find("Judgement").SendMessage("Perfect");
   }
   PlayerPrefs.SetInt("notecount", PlayerPrefs.GetInt("notecount")+1);
 }
 public void ResetCombo()
{
   PlayerPrefs.SetInt("GrooveGauge",   PlayerPrefs.GetInt("GrooveGauge")-4);
   combo = 0;
   UpdateGUI ();
}
public void win()
{
   PlayerPrefs.SetInt("Start", 0);
  SceneManager.LoadScene (4);
}
void GameOver()
{
   PlayerPrefs.SetInt("Start", 0);
   SceneManager.LoadScene(6);
}
void UpdateGUI()
{
  PlayerPrefs.SetInt ("Combo", combo);
  PlayerPrefs.SetInt ("Gauge",    ` PlayerPrefs.GetInt ("GrooveGauge") - 2);
```

```
    }
    public void Addscore()
    {
        int adder;
        if (accuracylevel == 1)
        {
          adder = 50;
        }
        else if (accuracylevel == 2)
        {
          adder=  75;
        }
        else
          adder =  100;
        PlayerPrefs.SetInt("Score", PlayerPrefs.GetInt("Score") + adder);
}
```

4. `Initiator.cs`

```
public class intantiator : MonoBehaviour {
  // Use this for initialization
  void Start () {
     GameObject Button = Resources.Load("SongLabel") as GameObject;
     string filePath = Application.dataPath+"/Resources/Midi Notes";
     DirectoryInfo f = new DirectoryInfo(filePath);
     foreach (var line in f.GetFiles("*.bytes"))
     {
         GameObject Par;
         Par = GameObject.Find("GridWidthElement");

         GameObject butt = Instantiate(Button);
         butt.name = line.Name.Replace(".bytes", "");
         butt.transform.SetParent(Par.transform, false);
         butt.GetComponentInChildren<Text>().text =
         line.Name.Replace(".bytes", "");
      }
   }

   // Update is called once per frame
    void Update () {
    }
  }
```

5. ButtonFunction.cs

```
public class ButtonFunction : MonoBehaviour {
   //source code untuk berpindah kehalaman yang lain
   public void LoadScene(int a)
   {
      if(a == 2|| a== 5)
      {
```

```
            emptythesongs();
        }
        SceneManager.LoadScene(a);
    }
    //mengosongkan lagu
    void emptythesongs()
    {
        PlayerPrefs.SetString("songName", "");
        PlayerPrefs.SetString("songBPM", "");
        PlayerPrefs.SetString("songLevel", "");
        PlayerPrefs.SetString("Custom.ini", "");
        PlayerPrefs.SetString("Custom.bytes", "");
        PlayerPrefs.SetString("Custom.wav", "");
    }
    public void Quit()
    {
        Application.Quit();
    }
}
```

6. musicinfosetter.cs

```
public class musicinfosetter : MonoBehaviour
{
string name;
bool instantiated = false;
private void Start()
{
    instantiated = false;
}
public void setMusicInfo()
{
    name = EventSystem.current.currentSelectedGameObject.name;
    PlayerPrefs.SetString("songName", name);
    string filePath = Application.dataPath + "/Resources/MidiConfig/" + name +
     ".ini";
    string config = File.ReadAllLines(filePath).First();
    string[] entries = config.Split('|');
    if (entries[0] == name)
    {
        string[] bpmFixed = entries[1].Split('.');
        PlayerPrefs.SetString("songBPM", "bpm: " + bpmFixed[0]);
        PlayerPrefs.SetString("songLevel","tingkat kesulitan: " + entries[5]);
    }
    //setelah itu instantiate tombol untuk start
    if(!instantiated)
    {
        GameObject ParentCanvas = GameObject.Find("Canvas");
        GameObject button;
        button = Instantiate(Resources.Load("TombolMulai")) as GameObject;
        button.transform.SetParent(ParentCanvas.transform, false);
```

```
    instantiated = true;
}
}
}
```

7. musicinfosetterforcustoms.cs

```
public class musicinfosetterforcustoms : MonoBehaviour
{
  string name;
  string filename, SourceFile, destFile;
  string namewithtype;
  string inifile, bytesfile, wavfile;
  bool instantiated = false;
  string filepath;
  private void Start()
  {
     PlayerPrefs.SetString("stats", "permainan dapat dimulai apabila data sudah
      lengkap");
     instantiated = false;
  }

  public void Update()
  {
    inifile = PlayerPrefs.GetString("Custom.ini");
    bytesfile = PlayerPrefs.GetString("Custom.bytes");
    wavfile = PlayerPrefs.GetString("Custom.wav");
    if (inifile != "")
     {
        filename = inifile.Split('\\').Last();
        SourceFile = inifile;
        destFile = System.IO.Path.Combine(Application.dataPath +
        "/Resources/MidiConfig", filename);
        System.IO.File.Copy(SourceFile, destFile, true);
     }
     if (bytesfile != "" && inifile != "" && wavfile != "" && !instantiated)
     {
         PlayerPrefs.SetString("stats", "Game sudah bisa dimulai");
         GameObject ParentCanvas = GameObject.Find("Canvas");
         GameObject button;
          button = Instantiate   (Resources.Load("TombolMulai")) as
         GameObject;
          button.transform.SetParent(ParentCanvas.transform, false);
          instantiated = true;
       }
     }
  }
  public void setSongInfo()
  {
```

```
            filepath = PlayerPrefs.GetString("Custom.ini");
            string config = File.ReadAllLines(filepath).First();
            string[] entries = config.Split('|');
            string[] bpmFixed = entries[1].Split('.');
            PlayerPrefs.SetString("songBPM", "bpm: " + bpmFixed[0]);
            PlayerPrefs.SetString("songLevel", "tingkat kesulitan: " + entries[5]);
        }
    public void SetName()
    {
        namewithtype = wavfile.Split('\\').Last();
        name = namewithtype.Replace(".wav","");
        PlayerPrefs.SetString("songName", name);
    }
}
```

LAMPIRAN 2

Contoh file.ini dari 10 lagu yang sudah di sebutkan di **Subbab 5.3**

1. cingcangkeling.ini

cingcangkeling|124.00|1.5|150|0|5|0|none|none

2. amparamparpisang.ini

amparamparpisang|140.00|1|145|1|5|0|none|none

3. cublakcublaksueng.ini

cublakcublaksueng|92.00|0.4|146|1|2|0|none|none

4. manukdadali.ini

manukdadali|103.00|2.2|147|1|2|0.2|none|none

5. oinenikeke.ini

oinanikeke|100.00|2.3|144|1|1|0|none|none

6. cintaterlarang.ini

cintaterlarang |140.00|2.3|147,152|1|3|0|none|none

7. akumalutapimau.ini

akumalutapimau|113.00|2.3|144,147|1|3|0|none|none

8. jikacintadia.ini

jikacintadia|160.00|2.3|147|1|3|0|none|none

9. jadiapayangkuinginkan.ini

jadiapayangkauinginkan|142.00|2.4|147|1|3|0|none|none

10. Adaapadenganmu.ini

adaapadenganmu|113.00|2.3|144,147,157|1|6|0|147,157|144

LAMPIRAN 3

Berikut Jawaban para Reponden untuk uji Pengguna pada Subbab 5.4

Responden: Dzulkarnain				
No.	**Nama Pengujian**		**Nilai**	**Rata-rata**
1.	Penilaian Antarmuka	Kemudahan Penggunaan	5	**3.8**
		Responsitifas UI	3	
		Keindahan Tampilan	4	
		Kesesuaian tema	4	
		Kelengkapan UI	3	
2	Performa Sistem permainan	Kelancaran Kinerja Aplikasi	5	**4,66**
		Kesesuaian *timing note* pada lagu	5	
		Kesesuaian Input dan Output	4	
3	Penilaian Konten	Kepantasan/Appropriasi Konten	5	**4**
		Ketertarikan terhadap konten	3	
4	Penilaian Desain Game	Variasi Tingkat Kesulitan	4	**4**
		Keseimbangan kondisi menang/kalah	5	
		Kesesuaian input dan output	3	

Responden: Arfian Fidiantoro				
No.	**Nama Pengujian**		**Nilai**	**Rata-rata**
1.	Penilaian Antarmuka	Kemudahan Penggunaan	4	**3.8**
		Responsitifas UI	4	
		Keindahan Tampilan	4	
		Kesesuaian tema	3	
		Kelengkapan UI	4	
2	Performa Sistem permainan	Kelancaran Kinerja Aplikasi	4	**3.66**
		Kesesuaian *timing note* pada lagu	4	
		Kesesuaian Input dan Output	3	
3	Penilaian Konten	Kepantasan/Appropriasi Konten	4	**4**
		Ketertarikan terhadap konten	4	
4	Penilaian Desain Game	Variasi Tingkat Kesulitan	3	**3**
		Keseimbangan kondisi menang/kalah	3	
		Kesesuaian input dan output	3	

Responden: Joshua Kevin Rahmadi				
No.	**Nama Pengujian**		**Nilai**	**Rata-rata**
1.	Penilaian Antarmuka	Kemudahan Penggunaan	3	**3.2**
		Responsitifas UI	2	
		Keindahan Tampilan	4	
		Kesesuaian tema	5	
		Kelengkapan UI	2	
2	Performa Sistem permainan	Kelancaran Kinerja Aplikasi	5	**4.66**
		Kesesuaian *timing note* pada lagu	4	
		Kesesuaian Input dan Output	5	
3	Penilaian Konten	Kepantasan/Appropriasi Konten	5	**4.5**
		Ketertarikan terhadap konten	4	
4	Penilaian Desain Game	Variasi Tingkat Kesulitan	3	**3.33**
		Keseimbangan kondisi menang/kalah	3	
		Kesesuaian input dan output	4	

Responden: Richardo Tiono				
No.	**Nama Pengujian**		**Nilai**	**Rata-rata**
1.	Penilaian Antarmuka	Kemudahan Penggunaan	2	**2.8**
		Responsitifas UI	4	
		Keindahan Tampilan	1	
		Kesesuaian tema	5	
		Kelengkapan UI	2	
2	Performa Sistem permainan	Kelancaran Kinerja Aplikasi	4	**4.66**
		Kesesuaian *timing note* pada lagu	5	
		Kesesuaian Input dan Output	5	
3	Penilaian Konten	Kepantasan/Appropriasi Konten	4	**3**
		Ketertarikan terhadap konten	2	
4	Penilaian Desain Game	Variasi Tingkat Kesulitan	5	**4**
		Keseimbangan kondisi menang/kalah	2	
		Kesesuaian input dan output	5	

Responden: Rifqi Maula Iqbal				
No.	**Nama Pengujian**		**Nilai**	**Rata-rata**
1.	Penilaian Antarmuka	Kemudahan Penggunaan	3	**2.8**
		Responsitifas UI	4	
		Keindahan Tampilan	3	
		Kesesuaian tema	4	
		Kelengkapan UI	5	
2	Performa Sistem permainan	Kelancaran Kinerja Aplikasi	4	**4.33**
		Kesesuaian *timing note* pada lagu	4	
		Kesesuaian Input dan Output	5	
3	Penilaian Konten	Kepantasan/Appropriasi Konten	4	**3.5**
		Ketertarikan terhadap konten	3	
4	Penilaian Desain Game	Variasi Tingkat Kesulitan	5	**4**
		Keseimbangan kondisi menang/kalah	2	
		Kesesuaian input dan output	5	

Responden: Monica Indah Hapsari				
No.	**Nama Pengujian**		**Nilai**	**Rata-rata**
1.	Penilaian Antarmuka	Kemudahan Penggunaan	3	**4**
		Responsitifas UI	4	
		Keindahan Tampilan	4	
		Kesesuaian tema	5	
		Kelengkapan UI	4	
2	Performa Sistem permainan	Kelancaran Kinerja Aplikasi	4	**4.66**
		Kesesuaian *timing note* pada lagu	5	
		Kesesuaian Input dan Output	5	
3	Penilaian Konten	Kepantasan/Appropriasi Konten	5	**4**
		Ketertarikan terhadap konten	3	
4	Penilaian Desain Game	Variasi Tingkat Kesulitan	5	**4.66**
		Keseimbangan kondisi menang/kalah	4	
		Kesesuaian input dan output	5	

BIODATA PENULIS

Penulis dilahirkan di Surabaya, 19 April 1995, merupakan anak pertama dari tiga bersaudara. Penulis telah menempuh pendidikan formal yaitu TK Rhapshody Surabaya (1999-2001), SD MIMI (2001-2007), SMPK Angelus Custos (2007-2010), SMAK Frateran Surabaya (2010-2013) dan mahasiswa S1 Departemen Informatika Institut Teknologi Sepuluh Nopember Surabaya rumpun mata kuliah Interaksi, Grafika dan Seni (2013-2017). Selama menjadi mahasiswa, Penulis yang memiliki hobi bermain *game* ,travelling dan sastra. kegiatan diluar akademik yang pernah diikuti adalah UKM IFLS (2014-2017). Penulis dapat dihubungi melalui *surel* stanley13@mhs.if.is.ac.id,dan id line renmod.